ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΑΣ, ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΚΑΙ ΤΗΝ ΤΕΧΝΟΛΟΓΙΑ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΥΚΤΙΑΚΟΥ ΠΑΙΧΝΙΔΙΟΥ «GEEKY-SNAKE»

ΒΑΣΙΛΕΙΟΣ ΓΕΝΝΑΡΗΣ
(ΑΜ: 1102)

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ
ΛΕΠΟΥΡΑΣ ΓΕΩΡΓΙΟΣ
ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ

Τρίπολη, Φεβρουάριος 2015

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω όλους όσους συνετέλεσαν στην εκπόνηση της διπλωματικής μου εργασίας.

- Την οικογένεια μου που μου έδωσε την δυνατότητα να σπουδάσω και να φέρω εις πέρας αυτόν τον μεταπτυχιακό τίτλο
- Τον καθηγητή μου, κύριο Γεώργιο Λέπουρα, για την υπομονή του καθόλη την διάρκεια, για την εμπιστοσύνη που μου έδειξε αναθέτοντάς μου τη διπλωματική αυτή εργασία, αλλά και για την καθοδήγηση και την παροχή συμβουλών καθ' όλη την διάρκεια της υλοποίησης της και επίσης την καλή του διάθεση σε όλη τη διάρκεια της προσπάθειάς μου.
- Τους φίλους μου για την κατανόηση τους και την συμβολή τους για να υλοποιηθεί αυτό το διασκεδαστικό διαδικτυακό παιχνίδι.

Πίνακας περιεχομένων

Περίληψη

Η βιομηχανία του Online Game Development αλλά και του Game Development γενικότερα είναι μια από τις μεγαλύτερες βιομηχανίες που δραστηριοποιούνται στο κομμάτι της τεχνολογίας των ηλεκτρονικών υπολογιστών και στην ανάπτυξη λογισμικού, αυτό έχει σαν αποτέλεσμα την ταχεία δημιουργία νέων απαιτήσεων από την αγορά και παράλληλα δημιουργεί την ανάγκη για την ύπαρξη νέων τεχνολογιών, όσον αφορά το software αλλά και το hardware. Ο βασικός λόγος για τον οποίο έχουν επενδυθεί τόσα χρήματα στην έρευνα αλλά και στην παραγωγή ηλεκτρονικών παιχνιδιών, είναι η τεράστια ζήτηση η οποία συνεχώς αυξάνεται όσο οι άνθρωποι εξοικειώνονται περισσότερο με τους υπολογιστές και η τεχνολογία γίνεται οικονομικά πιο προσιτή από ότι τα παλαιότερα χρόνια. Η εξέλιξη που υπάρχει τα τελευταία 15 χρόνια στον κλάδο αυτόν είναι ραγδαία, και έχουν δημιουργηθεί τεράστιες εταιρίες παγκόσμιος με χιλιάδες άτομα προσωπικό οι οποίες ασχολούνται μόνο με αυτό το αντικείμενο.

Ο σχεδιασμός και η δημιουργία παιχνιδιών δεν είναι ένα εύκολο θέμα για να μπορέσουμε να αναφερθούμε σε αυτό διεξοδικά. Για την πλήρη κατανόηση της διαδικασίας δημιουργίας ενός ηλεκτρονικού παιχνιδιού απαιτείται η κατανόηση μιας πολύπλοκης δομής που εμπεριέχει το δημιουργικό κομμάτι, το ψυχολογικό, το κομμάτι της τέχνης και της τεχνολογίας αλλά και το επιχειρηματικό. Αλλάζοντας τα δεδομένα σε μια από τις προηγούμενες παραμέτρους ταυτόχρονα επηρεάζονται και όλες οι υπόλοιπες. Σαν συμπέρασμα κάποιος ο οποίος θέλει να σχεδιάσει και να υλοποιήσει ένα παιχνίδι θα πρέπει να έχει υπόψη του τις προηγούμενες παραμέτρους που συντελούν στην δομή ενός παιχνιδιού και να κατανοεί πως επηρεάζει η μία την άλλη.

Από εμάς έγινε μια προσπάθεια, για τις ανάγκες της διπλωματικής εργασίας του μεταπτυχιακού με τίτλο «Συστήματα Λογισμικού» , να σχεδιαστεί και να υλοποιηθεί ένα παιχνίδι το οποίο θα έχει πρωτότυπο σενάριο, θα είναι εύκολα κατανοητό από τον μέσο χρήστη, θα δίνεται η δυνατότητα να συμμετέχουν από δύο μέχρι τέσσερις χρήστες, θα δημιουργεί στον παίκτη μια αίσθηση ανταγωνισμού και θα τον προκαλεί να σκεφτεί με στρατηγικό τρόπο για να επιφέρει το επιθυμητό αποτέλεσμα το οποίο δεν είναι άλλο από την νίκη. Το παιχνίδι που τελικώς δημιουργήσαμε είναι ένα επιτραπέζιο παιχνίδι καρτών, σε ηλεκτρονική μορφή, το οποίο έχει πολλά στοιχεία από το επιτραπέζιο, που οι περισσότεροι θυμόμαστε από τα παιδικά μας χρόνια «Φιδάκι και Σκάλες», και είναι εμπλουτισμένο με κάρτες ενεργειών, τις οποίες ο χρήστης μπορεί να τις χρησιμοποιήσει κατά την διάρκεια του παιχνιδιού προς όφελος του, επίσης το παιχνίδι διαθέτει στα περισσότερα τετράγωνα του ταμπλό του ερωτήσεις, στις οποίες απαντώντας σωστά τον βοηθούν θετικά στην βαθμολογία του. Η προσθήκη των ερωτήσεων προσθέτει ένα μαθησιακό χαρακτήρα στο παιχνίδι, ο οποίος κινείται στο πεδίο της πληροφορικής, και οι κάρτες ενεργειών οι οποίες υποκινούν τον χρήστη να αναπτύξει στρατηγική σκέψη για την χρήση τους με σκοπό το μεγαλύτερο όφελος της καθεμίας στην τελική βαθμολογία του.

Abstract

The industry of Online Game Development and Game Development in general, is one of the largest industries operating in the field of electronic computer technology and software development. This results in the rapid creation of new requirements in the market and also gives rise to the need for developing new technologies regarding the software and the hardware. The main reason for investing such large amounts of money in the research and production of computer games is the huge demand which is continually growing, as people become more familiar with computers and technology becomes more affordable than in the previous years. The development in the industry the last 15 years is rapid and has resulted in the creation of global corporations with thousands of employees who deal exclusively with this sector.

Designing and developing a computer game is not an easy subject to analyze in detail. To fully understand the process of creating a computer game requires the understanding of a complex structure which contains the creative part, the psychological, the artistic and technological part as well as the business know-how. Altering the data in one of the preceding parameters has an impact on all others as well. In conclusion, someone who wants to design and implement a game should be aware of the preceding parameters which contribute to the structure of a game and should understand how they interact with each other.

This paper is an attempt, for the needs of the graduate thesis entitled "Software Systems", to design and implement a game that has an original scenario, it is easily understood by the average user, two to four players can participate and causes the player a sense of competition as well as challenges them to think strategically in order to achieve the desired result, which is nothing other than to win.
The game finally created is a board game of cards, in electronic form, which includes many elements from the classic board game "Snakes and Ladders', which most might remember from our childhood. The game is enriched with action cards, which the user can use during the game in their benefit. Also the game features a set of questions in most blocks of the dashboard, to which the right answer helps positively to the score.

The addition of the questions imparts a learning character to the game, which moves in the field of information technology and finally the action cards incite the user to think strategically on how to use them, in order to achieve a higher final score.

Εισαγωγή

Το παιχνίδι που δημιουργήθηκε με σκοπό της εκπόνησης της διπλωματική εργασίας, για το μεταπτυχιακό ειδίκευσης στην Επιστήμη και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πελοποννήσου του τμήματος Πληροφορικής και Τηλεπικοινωνιών, είναι η εξέλιξη και ταυτόχρονα μεταφορά σε ηλεκτρονική-διαδικτυακή μορφή ενός υπάρχοντος επιτραπέζιου παιχνιδιού το οποίο είναι γνωστό ως «Φιδάκι και Σκάλες» και το οποίο οι περισσότεροι γνωρίζουμε από τα παιδικά μας χρόνια. Σκοπός ήταν – και με δεδομένο ότι το παιχνίδι αυτό απευθύνεται σε αρκετά μικρές ηλικίες, να μπορέσουμε να το μετατρέψουμε και να το εξελίξουμε σε ένα παιχνίδι ακόμη πιο ελκυστικό και ενδιαφέρον και για κοινό μεγαλύτερων ηλικιών προσθέτοντας νέα στοιχεία κατά την διάρκεια του, με κύριο σκοπό την δημιουργία αισθήματος χαράς και διασκέδασης. Προς αυτήν την κατεύθυνση εξετάσαμε άλλα επιτραπέζια παιγνίδια που απευθύνονται σε μεγαλύτερες ηλικίες παικτών επιλέγοντας στοιχεία τα οποία θα μπορούσαν να κάνουν την πλοκή του παιγνιδιού πιο ενδιαφέρουσα. Ο στόχος ήταν η δημιουργία, στον παίκτη, αισθήματος αυξημένης αγωνίας μέσα από την προσθήκη κανόνων που τροποποιούν την κίνηση πάνω στο ταμπλό και παράλληλα εισάγουν την ανάγκη σχεδιασμού στρατηγικής από τη μεριά του. Σε αντιπαραβολή το κλασσικό παιχνίδι «Φιδάκι και Σκάλες» βασίζεται μόνο στον παράγοντα της τύχης για τον τρόπο κίνησης των παικτών κατά την διάρκεια του και για το ποιος θα είναι τελικά ο νικητής. Στην εξελιγμένη του έκδοση το παιχνίδι δε βασίζεται εξ' ολοκλήρου στην τύχη αλλά πλέον σημαντικό ρόλο για την νικητήρια έκβαση έχουν οι γνώσεις αλλά και η κρίση του εκάστοτε παίκτη. Το παιχνίδι μας είναι μετρίου τύπου δυσκολίας και απευθύνεται σε ηλικίες 15+ λόγο κυρίως των ερωτήσεων του. Παράλληλα έγινε μια προσπάθεια ώστε να μην χαθεί ο τόνος του παιχνιδιού που μας θυμίζει τα παιδικά μας χρόνια και εικονικά τουλάχιστον να μας παραμείνει γνώριμο και ελκυστικό.

Έρευνα

Στο βιβλίο του Jesse Schell, ο οποίος είναι ένας από τους πλέον γνωστούς σχεδιαστές βίντεο παιγνιδιών και καθηγητής στο πανεπιστήμιο Carnegie Mellon της Αμερικής στο αντικείμενο του σχεδιασμού παιχνιδιών, με τίτλο "The Art of Game Design" αναφέρει το ακόλουθο:

> "Game design is the act of deciding what a game should be. That's it. On the surface, it sounds too simple". Στα ελληνικά το παραπάνω σημαίνει ότι "Ο σχεδιασμός παιχνιδιών είναι η επιλογή του πώς θα πρέπει να είναι ένα παιχνίδι. Αυτό μόνο. Επιφανειακά, ακούγεται τόσο απλό." (Schell, 2008) (Wikipwdia, 2015)

Η τέχνη της σχεδίασης και δημιουργίας παιχνιδιών είναι αρκετά περίπλοκη και δύσκολη. Απαιτείται πολύ καλή γνώση του αντικείμενου, δημιουργική σκέψη και παράλληλα σε αυτήν να συνυπάρχουν όλες οι παράμετροι που κάνουν ένα παιχνίδι επιτυχημένο από το σενάριό

του μέχρι τον σχεδιασμό του. Επίσης ένα πολύ σημαντικό κομμάτι του σχεδιασμού είναι να έχει γίνει κατανοητό πιο θα είναι το κοινό στο οποίο θα απευθύνεται το παιχνίδι, παράγοντες όπως η ηλικία, η καταγωγή αλλά επίσης και ο σκοπός του παιχνιδιού, αν δηλαδή αυτός είναι ψυχαγωγικός, εκπαιδευτικός ή κάποιος άλλος είναι απαραίτητο να είναι απόλυτος ξεκάθαροι πριν ξεκινήσει ο σχεδιασμός.

Η τελική απόφαση για την δημιουργία του συγκεκριμένου παιχνιδιού πάρθηκε μετά από αρκετή σκέψη, ανάλυση και έρευνα πάνω σε ήδη υπάρχοντα παιχνίδια, ώστε να καταφέρουμε να δημιουργήσουμε ένα πρωτότυπο σενάριο το οποίο θα είναι ελκυστικό και θα δημιουργήσει στον παίκτη αίσθημα χαράς, φιλίας, εκπλήρωσης στόχου και πολλά ακόμη τα οποία είναι απαραίτητα στην σύνθεση για την δημιουργία ενός καλού παιχνιδιού. Επιπλέον πραγματοποιήθηκε έρευνα και ετοιμάστηκαν αρκετά προσχέδια τόσο για τον τελικό σχεδιασμό του παιχνιδιού αλλά και για την ακολουθία που θα πρέπει κάποιος χρήστης να ακολουθήσει για να παίξει και να δημιουργήσει ένα παιχνίδι, τα οποία θα αναλύσουμε στα επόμενα κεφάλαια.

Επίσης έγινε αρκετά μεγάλη προσπάθεια ώστε ο χρήστης ο οποίος θα μπει στο παιχνίδι μας και θα παίξει να μην συναντήσει κάποιο στοιχείο το οποίο θα τον αποτρέψει και θα τον δυσκολέψει από την αρχή που θα επισκεφτεί την σελίδα του παιχνιδιού μας μέχρι και την ολοκλήρωση αυτού, με σκοπό να αποφευχθεί οποιαδήποτε αρνητική εντύπωση η οποία θα είχε ως αποτέλεσμα την αποτροπή του από το να ξανά επισκεφτεί το παιχνίδι μας και γενικότερα θα δημιουργούσε αρνητικές κριτικές γι αυτό.

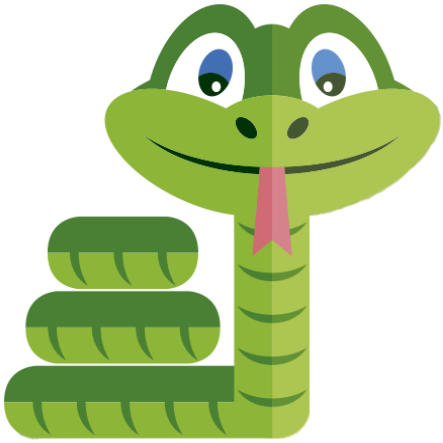
Αυτό για να επιτευχθεί ακολουθήσαμε μια διαδικασία που μας έδινε αμέσως ανατροφοδότηση πληροφορίας, δηλαδή με το που προσθέταμε κάτι καινούργιο ή αλλάζαμε κάτι που ήδη υπήρχε στο παιχνίδι το ελέγχαμε είτε μόνοι μας είτε παρέα με φίλους ώστε να μας ενημερώνουν για το τι τους άρεσε και τι όχι πάνω στην αλλαγή ή στην προσθήκη. Αυτή ήταν η διαδικασία που ακολουθήθηκε και στον σχεδιασμό αλλά και στην τεχνική υλοποίηση.

Τίτλος – Λογότυπο

Το όνομα του παιχνιδιού μας είναι Geeky-Snake, το οποίο γεννήθηκε ως ιδέα από τους δύο παρακάτω λόγους. Ο πρώτος είναι γιατί το παιχνίδι είναι μεταφορά – εξέλιξη του παιχνιδιού "Snakes and Ladders" (Φιδάκι και Σκάλες), όπου οι περισσότεροι γνωρίζουμε από τα παιδικά μας χρόνια, και θέλαμε το όνομα του να μας παραπέμπει σε κάτι γνώριμο, και ο δεύτερος λόγος, αναφερόμενοι στο δεύτερο μισό του τίτλου το "Geeky" είναι γιατί οι ερωτήσεις που περιέχονται στο παιχνίδι έχουν ως θεματολογία την πληροφορική και τον προγραμματισμό και ο χαρακτηρισμός "Geek" αποδίδεται χαριτολογώντας συχνά σε ανθρώπους που ασχολούνται με τον κλάδο αυτόν. Η *εικόνα 1* είναι η παρουσίαση του ονόματος στην αρχική σελίδα του παιχνιδιού.

Εικόνα 1. Τίτλος παιγνιδιού

Το λογότυπο του παιχνιδιού είναι το φιδάκι που βλέπουμε στην *εικόνα 2* το οποίο είναι σχεδιασμένο σε σκιτσογραφική μορφή (cartoon) κατάλληλη για το παιχνίδι και το ύφος το οποίο θέλουμε να αποδώσουμε. Μορφή η οποία θέλουμε να μας παραπέμπει σε κάτι το οποίο προσφέρει διασκέδαση και χαρά και σύμφωνα με τις γνώμες των ανθρώπων που ρωτήσαμε αν θα τους παρέπεμπε σε ένα παιχνίδι και τι τύπου θα περίμεναν να είναι αυτό, ήταν θετικές προς τον στόχο μας.



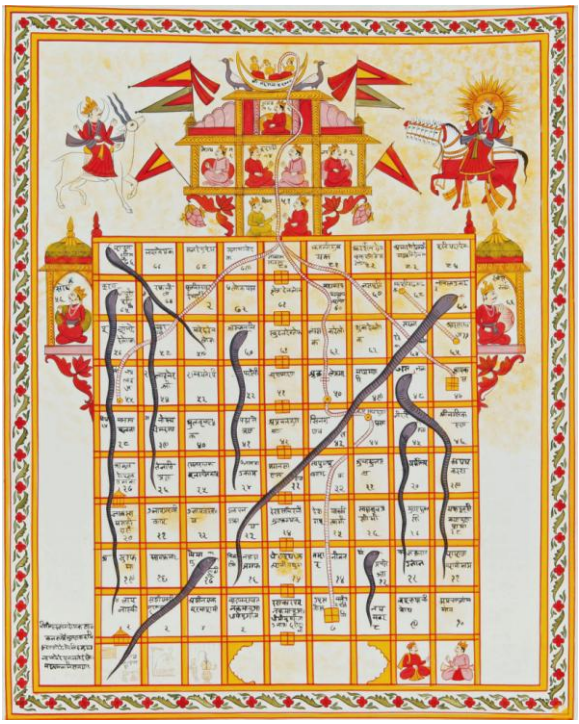
Εικόνα 2. Λογότυπο παιγνιδιού

Η έκδοση του παιχνιδιού η οποία υλοποιήθηκε είναι η πρώτη (v.1.0) και πιθανότατα να υπάρξει εξέλιξη πάνω στην ήδη υπάρχουσα.

Ιστορική αναδρομή και εξέλιξη του παιχνιδιού

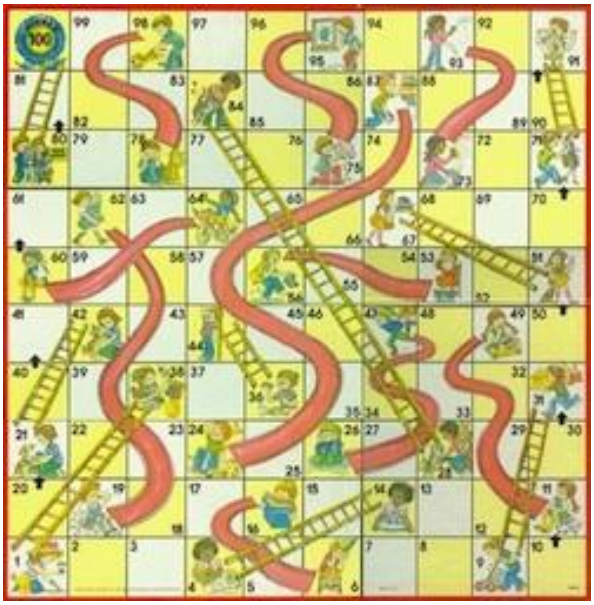
Το παιχνίδι μας με όνομα Geeky-Snake δημιουργήθηκε με σκοπό την ψυχαγωγία την εκμάθηση αλλά και την παρακίνηση για την δημιουργία στρατηγικής σκέψης για να μπορέσει ο εκάστοτε παίκτης να ολοκληρώσει το παιχνίδι επιτυχώς. Το επιτραπέζιο πάνω στο οποίο είναι βασισμένο το παιχνίδι μας προέρχεται από την Ινδία και υπάρχει από αρχαιοτάτων χρόνων με ελάχιστα διαφορετική μορφή και ακριβώς ίδια πλοκή με το σημερινό επιτραπέζιο. Είναι συνυφασμένο με την Ινδική φιλοσοφία οπού η κίνηση του παίκτη πάνω στο ταμπλό συνδυάζεται μεταφορικά με το ταξίδι της ζωής το οποίο εμπεριέχει καλές (σκάλες) και κακές στιγμές (φίδια). Επίσης στην αρχαία Ινδία το παιχνίδι αυτό, μεταφορικά ήταν συνδυασμένο

με την μοίρα και το κάρμα για τον κάθε παίκτη όποτε αν ήταν τυχερός κάποιος και ήταν γραφτό του θα κέρδιζε το παιχνίδι χρησιμοποιώντας όλα τα δυνατά μέσα που παρέχονται, κάτι αντίστοιχο συμβαίνει και στην ζωή σύμφωνα με κάποιες φιλοσοφίες της Ινδίας, δηλαδή το αν είναι γραφτό να φτάσεις "ψηλά" ή αν θα σου πηγαίνουν καλά τα πράγματα στην ζωή σου είναι θέμα της μοίρας. Στην *εικόνα 3* βλέπουμε μια μορφή του παιχνιδιού όπως υπήρχε τον 19 αιώνα στην Ινδία. (Wikipedia, 2015)



Εικόνα 3. Παλιό ταμπλό παιγνιδιού «Φιδάκι»

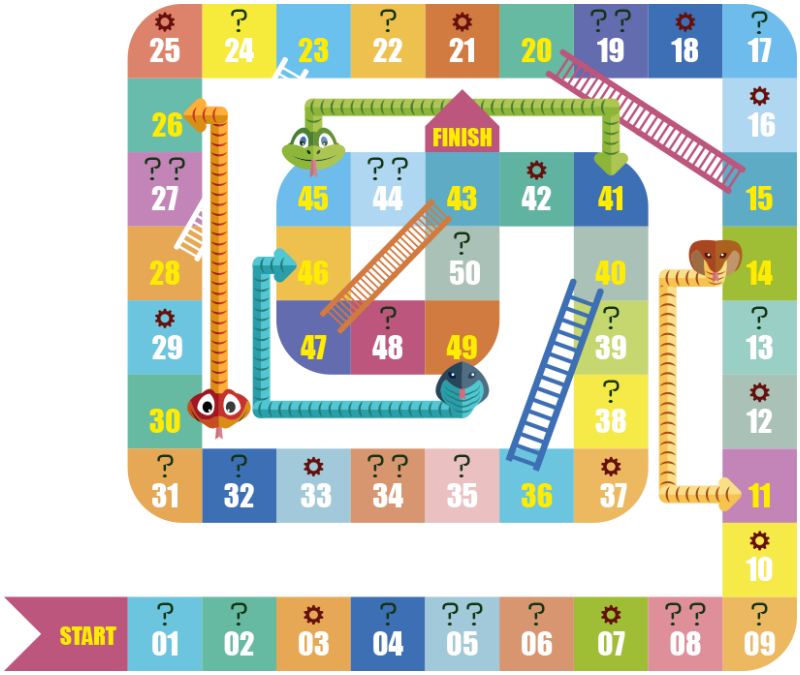
Το παιχνίδι με την σημερινή του μορφή όπως το γνωρίζουμε ως ένα επιτραπέζιο παιχνίδι τύχης χωρίς κάποια ιδιαίτερη φιλοσοφία ή χωρίς να απαιτεί κάποιες ειδικές γνώσεις για να μπορέσει κάποιος να παίζει πρωτοεμφανίστηκε το 1943 από τον πρωτοπόρο στην βιομηχανία των παιχνιδιών Milton Bradley, το οποίο από την πρώτη στιγμή απευθυνόταν σε μικρές ηλικίες. Στην *εικόνα 4* βλέπουμε μία κλασσική μορφή του παιχνιδιού όταν πρωτοεμφανίστηκε στις δυτικές χώρες.



Εικόνα 4. Σύγχρονη εκδοχή του ταμπλό

Εμείς ως σκοπό είχαμε να εξελίξουμε την λογική του ήδη υπάρχοντος παιχνιδιού ώστε να γίνει πιο ελκυστικό και ενδιαφέρων και για τις μεγαλύτερες ηλικίες με τρόπο τέτοιον ώστε ο χρήστης να πρέπει να χρησιμοποιήσει το μυαλό και τις γνώσεις του με σκοπό την νίκη του παιχνιδιού.

Η πρώτη πολύ σημαντική αλλαγή προς αυτήν την κατεύθυνση είναι στο ταμπλό του παιχνιδιού το οποίο βλέπουμε στην *εικόνα 5*. Ο λόγος για τον οποίο καταλήξαμε σε αυτήν την επιλογή μετά από αρκετή σκέψη και προσχέδια, για τα οποία θα αναφερθούμε σε επόμενο κεφάλαιο, είναι γιατί η διαδρομή σε σχήμα κουλουριασμένου φιδιού είναι πρωτότυπη όσον αφορά το συγκεκριμένο παιχνίδι με αυτό να είναι συμπέρασμα έρευνας σε αρκετά μεγάλο εύρος του ίδιου ή και παραλλαγών του παιχνιδιού αυτού. Ταυτόχρονα αρκετά στοιχεία του σχηματισμού αυτού παραπέμπουν σε πολλά στοιχεία του παιχνιδιού μας κάτι το οποίο το έκανε ως την επικρατέστερη από τις ιδέες μας. Η αναβαθμισμένη εκδοχή του παιχνιδιού έχει πλέον ένα ταμπλό με 50 τετράγωνα πάνω στα οποία βρίσκονται τέσσερις σκάλες και τέσσερα φιδάκια. Επίσης πάνω σε κάθε τετράγωνο εκτός από αυτά που έχουν σκάλες και φίδια θα λειτουργούν κάποιες ενέργειες. Οι ενέργειες αυτές θα χωρίζονται σε δύο κατηγορίες, η πρώτη η οποία υπάρχει και στην πλειοψηφία των τετράγωνων είναι ερωτήσεις και η δεύτερη κατηγορία είναι κάποιες κάρτες ενεργειών τις οποίες θα μπορούν να χρησιμοποιούν κατά την διάρκεια του παιχνιδιού προς όφελος τους. Στην συνέχεια θα γίνει ανάλυση του σεναρίου του παιχνιδιού.



Εικόνα 5. Η προτεινόμενη εκδοχή του ταμπλό

Ανάλυση Παιχνιδιού

Το μέγεθος του ταμπλό του επιτραπέζιου κλασσικού παιχνιδιού ποικίλει από 8*8 , 10*10 ακόμη και 12*12 τετράγωνα, όπως επίσης ποικίλουν και τα φιδάκια και οι σκάλες που υπάρχουν πάνω σε αυτό. Τα φιδάκια και οι σκάλες χρησιμεύουν στην μετακίνηση του χρήστη πάνω στο ταμπλό. Δηλαδή αν κάποιος χρήστης τερματίσει τη ζαριά-κίνησή του πάνω σε ένα τετράγωνο το οποίο ξεκινάει κάποια σκάλα είναι υποχρεωμένος να την ακολουθήσει και να προχωρήσει πιο γρήγορα μέσα στο παιχνίδι, το ίδιο συμβαίνει και αν τερματίσει τη ζαριά-κίνησή του πάνω σε ένα τετράγωνο οπού έχει κάποιο φιδάκι το οποίο και αυτό είναι υποχρεωμένος να το ακολουθήσει αλλά αυτήν την φόρα με αρνητικό προς τον παίκτη αποτέλεσμα γιατί η ενεργεία αυτήν τον απομακρύνει από τον τερματισμό ενώ η σκάλα τον φέρνει πιο κοντά σε αυτόν. Το παιχνίδι παίζεται από παλαιά και μέχρι και σήμερα με ένα ζάρι (μπορεί να κινηθεί κάθε παίκτης από 1 μέχρι 6 τετράγωνα) και μπορούν να συμμετέχουν από δύο και πάνω παίκτες με διάρκεια παιχνιδιού από 15 μέχρι και 45 λεπτά εξαρτώμενο πάντα από τον αριθμό των παικτών αλλά και την ταχύτητα με την οποία κινούνται πάνω στο ταμπλό.

Ο τύπος του παιχνιδιού μας είναι παραδοσιακού επιτραπέζιου ερωτήσεων και αινιγμάτων το οποίο έχει αρκετά στοιχεία από παιχνίδια στρατηγικής και υπάρχει μόνο σε ηλεκτρονική διαδικτυακή μορφή. Το ύφος του παιχνιδιού μας είναι σκιτσογραφικό και πάνω σε αυτό υπάρχουν κάποια νέα στοιχεία τα οποία είναι ουσιαστικά και αυτά που το διαφοροποιούν από το κλασσικό επιτραπέζιο.

Σενάριο παιχνιδιού

Το ταμπλό του παιχνιδιού έχει μια διαδρομή η οποία σχεδιάστηκε σε μορφή κουλουριασμένου φιδιού και αποτελείται από 50 τετράγωνα. Στο παιχνίδι μπορούν να παίξουν από δύο έως και τέσσερις παίκτες χρησιμοποιώντας μόνο ένα ζάρι. Δεν έχει σημασία το ποιος θα ξεκινήσει πρώτος γιατί επίσης δεν έχει σημασία ποιος θα τερματίσει πρώτος. Ο νικητής βγαίνει με τον τερματισμό όλων των παικτών και κερδισμένος είναι αυτός με τους περισσότερους πόντους. Η σειρά με την οποία θα ξεκινήσουν το παιχνίδι οι παίκτες καθορίζεται από την σειρά με την οποία εισήχθησαν στο παιχνίδι. Κάθε ένα τετράγωνο για το οποίο ο χρήστης κινείται πάνω στο ταμπλό θα παίρνει και από έναν βαθμό. Η διάρκεια του παιχνιδιού θα κυμαίνεται από 20 λεπτά μέχρι και τη μια ώρα.

Σε κάθε τετράγωνό του ταμπλό εκτός αυτών των οποίων εφάπτονται σε κάποια σκάλα ή φιδάκι, υπάρχουν τα σύμβολά που βλέπουμε στην *εικόνα 6*.

Εικόνα 6. Σύμβολα του εξελιγμένου παιγνιδιού

Στην πλειοψηφία των τετραγώνων υπάρχει το σύμβολο του ερωτηματικού το όποιο είναι το σύμβολο για τις ερωτήσεις, αυτό σημαίνει ότι όταν ο χρήστης ολοκληρώσει την ζάρια του σε κάποιο από αυτά τα τετράγωνα θα εμφανιστεί μία ερώτηση προς απάντηση σε αυτόν. Οι ερωτήσεις αυτές θα είναι πολλαπλής επιλογής και θα αποδίδουν, αν απαντηθούν σωστά, βαθμούς στον παίκτη ο οποίος απάντησε. Όπως παρατηρούμε στην *εικόνα 5,* του ταμπλό του παιχνιδιού μας, σε κάποια από τα τετράγωνα των ερωτήσεων υπάρχουν διπλά ερωτηματικά, αυτό σημαίνει ότι υπάρχουν δύο κατηγορίες ερωτήσεων, οι εύκολη κατηγορία, όπου συμβολίζεται με το ένα ερωτηματικό, και οι δύσκολες οι οποίες συμβολίζονται με δύο ερωτηματικά, οι δύσκολες ερωτήσεις αποδίδουν διπλάσιους πόντους στον χρήστη που θα απαντήσει σωστά.

Όταν κάποιος χρήστης ολοκληρώσει την ζαριά του στην αρχή μιας σκάλας ή σε κάποιο φιδάκι του ταμπλό είναι υποχρεωμένος να το ακολουθήσει, όπως συνέβαινε και στο κλασσικό επιτραπέζιο, εκτός από κάποιες περιπτώσεις τις οποίες θα εξηγήσουμε αναλυτικά παρακάτω. Στην περίπτωση που ο χρήστης ακολουθήσει κάποια σκάλα ή φιδάκι θα του προστίθενται ή θα του αφαιρούνται αντίστοιχα δυο βαθμοί από την συνολική του βαθμολογία και όχι τόσοι όσοι είναι η διαφορά των τετραγώνων που μετακινήθηκε.

Παράδειγμα: αν ένας παίκτης έχει 22 βαθμούς και η ζαριά του τον μετακινήσει στο τετράγωνο με αριθμό 15 αμέσως αυτός θα μεταφερθεί στο τετράγωνο με αριθμό 20, λόγω της σκάλας η οποία ξεκινάει από το τετράγωνο με αριθμό 15, αυξάνοντας την βαθμολογία του κατά 2 βαθμούς, δηλαδή 24, αντί για 5 βαθμούς που θα κέρδιζε αν πήγαινε στο 20 μέσω των τετραγώνων χωρίς την χρήση της σκάλας.

Επίσης όποιος παίκτης ολοκληρώσει την ζαριά του σε τετράγωνο που υπάρχει το σύμβολο του γραναζιού θα του εμφανίζεται μια κάρτα η οποία θα του εξηγεί την δυνατότητα που θα του δίνεται μέσω της χρήσης της κάρτας να κάνει κάποιες ενέργειες προς όφελος του κατά την διάρκεια του παιχνιδιού. Οι κάρτες αυτές θα είναι διαθέσιμες στον παίκτη όταν θα έρχεται η σειρά και όταν θα μπορεί να εφαρμόσει κάποια από τις κερδισμένες αυτές κάρτες στην τρέχουσα θέση που βρίσκεται πάνω στο ταμπλό.

Αναλυτικά θα υπάρχουν οι παρακάτω επτά κατηγορίες καρτών που θα δίνουν τις εξής δυνατότητες:

- 6 κάρτες οι οποίες θα επιτρέπουν στον παίκτη να μην ακολουθήσει το φιδάκι.
- 6 κάρτες οι οποίες θα επιτρέπουν στον παίκτη να μην ακολουθήσει την σκάλα.
- 5 κάρτες οι οποίες θα επιτρέπουν στον παίκτη να προχωρήσει 3 τετράγωνα στο ταμπλό προσθέτοντας 3 πόντους στην βαθμολογία του.
- 5 κάρτες οι οποίες θα επιτρέπουν στον παίκτη να προχωρήσει 5 τετράγωνα στο ταμπλό προσθέτοντας 5 πόντους στην βαθμολογία του.
- 5 κάρτες οι οποίες θα επιτρέπουν στον παίκτη να προχωρήσει 3 τετράγωνα στο ταμπλό χωρίς να προστεθούν πόντοι στην βαθμολογία του.
- 5 κάρτες οι οποίες θα επιτρέπουν στον παίκτη να κινηθεί προς τα πίσω 3 τετράγωνα στο ταμπλό του παιχνιδιού χωρίς να αφαιρεθούν πόντοι στην βαθμολογία του.
- 5 κάρτες οι οποίες θα επιτρέπουν στον παίκτη να κινηθεί προς τα πίσω 3 τετράγωνα στο ταμπλό του παιχνιδιού αφαιρώντας 3 πόντους στην βαθμολογία του.
- 5 κάρτες οι οποίες θα επιτρέπουν στον παίκτη να κινηθεί προς τα πίσω 5 τετράγωνα στο ταμπλό του παιχνιδιού αφαιρώντας 5 πόντους στην βαθμολογία του.
- 6 κάρτες που θα διπλασιάζουν του πόντους μιας απάντησης. Η κάρτα θα πρέπει να χρησιμοποιείται πριν διαβάσει την ερώτηση ο παίκτης.

Θα αποφεύγεται οι παίκτες να παίρνουν ίδιου τύπου κάρτες κατά την διάρκεια ενός παιχνιδιού.

Επόμενος η τελική βαθμολογία θα βγαίνει σαν αποτέλεσμα από:

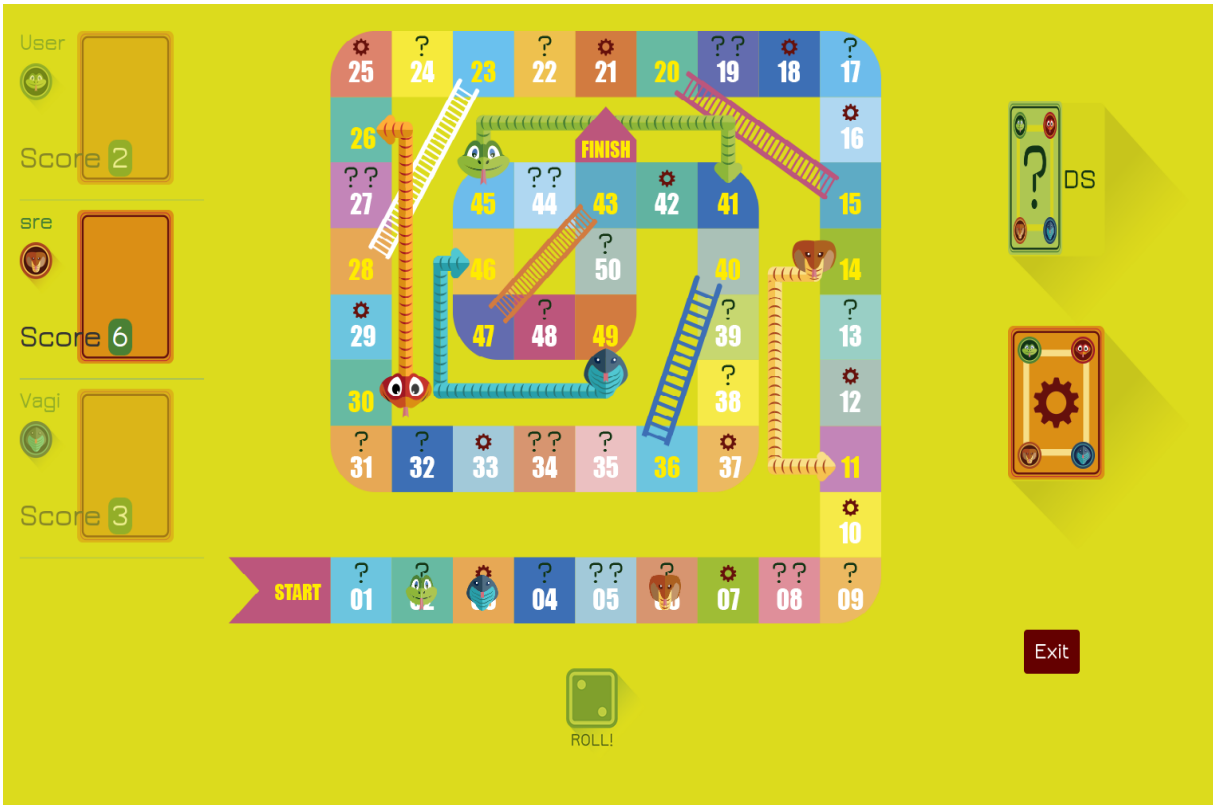
1. Κάθε τετράγωνο για το οποίο κινείται ο κάθε παίκτης θα παίρνει και από έναν βαθμό.
2. Κάθε εύκολη και σωστά απαντημένη ερώτηση θα δίνει 1 βαθμό.
3. Κάθε δύσκολη και σωστά απαντημένη ερώτηση θα δίνει 2 βαθμούς.
4. Η χρήση σκάλας θα δίνει στον παίκτη από 2 βαθμούς και όχι όσους βαθμούς θα του έδιναν τα τετράγωνα τα οποία μετακινήθηκε.
5. Η χρήση φιδιού θα αφαιρει από στον παίκτη από 2 βαθμούς και όχι όσους βαθμούς θα

του αφαιρούνταν από τα τετράγωνα τα οποία μετακινήθηκε προς τα πίσω.

6. Όταν γίνεται χρήση της κάρτα διπλασιασμού των βαθμών της ερώτησης και η ερώτηση απαντηθεί σωστά θα δίνει 2 βαθμούς για τις εύκολες ερωτήσεις και 4 για τις δύσκολες.
7. Τέλος σε περίπτωση χρήση κάρτας για μετακίνηση του παίκτη πάνω στο ταμπλό θα προστίθενται ή θα αφαιρούνται βαθμοί αναλόγως την κάρτα.
8. Σε περίπτωση ισοβαθμίας θα είναι παραπάνω από ένας οι νικητές.

Στην *εικόνα 7* παρατηρούμε ένα στιγμιότυπο του παιχνιδιού μας το οποίο είναι ένα παιχνίδι όπου την στιγμή της αποτύπωσης παιζόταν από τρείς παίκτες.

Κάθε παιχνίδι που θα δημιουργείται θα έχει κάποιον νεκρό χρόνο του μεγέθους του ενός λεπτού έτσι ώστε να δίνεται η δυνατότητα να προλάβουν όλοι οι παίκτες να μπουν στο παιχνίδι πριν αυτό ξεκινήσει. Κατά την διάρκεια του χρόνου αυτού οι παίκτες που έχουν ήδη μπει στο παιχνίδι δεν μπορούν να κάνουν καμία απολύτως ενέργεια γιατί όλες οι λειτουργίες είναι απενεργοποιημένες και ένας μετρητής δείχνει το υπόλοιπο του χρόνου μέχρι την έναρξη του παιχνιδιού. Με το που ξεκινήσει το παιχνίδι αφαιρείται από την λίστα με τα τρέχοντα παιχνίδια, στην σελίδα "Search Game" το κουμπί "Join" του συγκεκριμένου παιχνιδιού.



Εικόνα 7. Οθόνη παιγνιδιού

Ακολουθία παιχνιδιού

Για την δημιουργία, ανάπτυξη και ολοκλήρωση ενός παιχνιδιού χρειάζονται να ακολουθηθούν κάποια βήματα. Παρακάτω παρουσιάζεται μια βασική ακολουθία για την έναρξη και ολοκλήρωση ενός παιχνιδιού.

- Εισέρχεται ο παίκτης στην αρχική σελίδα του παιχνιδιού μας. Σε αυτήν υπάρχουν δύο κουμπιά ένα που έχει όνομα PLAY και οδηγεί τον παίκτη στην σελίδα δημιουργίας ενός νέου παιχνιδιού ή επιλογής ενός από τα ήδη υπάρχοντα και το κουμπί info το οποίο μας οδηγεί σε μία σελίδα οπού αναφέρονται οι κανόνες του παιχνιδιού.
- Αν ο παίκτης επιλέξει από την αρχική σελίδα να διαβάσει τους κανόνες του παιχνιδιού στην σελίδα Info, τότε σε αυτήν την σελίδα του δίνεται μόνο μία επιλογή, να γυρίσει στην αρχική σελίδα μέσω του κουμπιού Back.
- Αν ο χρήστης έχει πατήσει PLAY στην αρχική σελίδα και έχει βρεθεί στην σελίδα αναζήτησης ή δημιουργίας παιχνιδιού, τότε έχει τρεις επιλογές είτε να συμμετέχει σε κάποιο υπάρχον διαθέσιμο παιχνίδι είτε να δημιουργήσει το δικό του είτε να γυρίσει στην αρχική σελίδα μέσω του κουμπιού Back. Για την δημιουργία ενός νέου παιχνιδιού υπάρχει το κουμπί με όνομα Create. Όταν πατηθεί το κουμπί αυτό ανοίγει μια φόρμα στην οποία ο χρήστης θα πρέπει να συμπληρώσει το όνομα του παιχνιδιού το οποίο θέλει να δημιουργήσει, επίσης του δίνεται η επιλογή αν το παιχνίδι θα είναι ιδιωτικό, δηλαδή αν θα χρειάζεται κωδικός για να συμμετέχει κάποιος άλλος παίκτης, και τέλος αν είναι το πρώτο παιχνίδι που παίζει ο συγκεκριμένος χρήστης θα του ζητηθεί και όνομα στην ίδια φόρμα. Όταν συμπληρωθούν όλα τα στοιχεία και θέλουμε να ολοκληρώσουμε την δημιουργία του παιχνιδιού απλά ο παίκτης πρέπει να πατήσει το κουμπί Create της φόρμας για να μεταβεί στην σελίδα του παιχνιδιού. Επίσης δίνεται η δυνατότητα στους χρήστες να συμμετέχουν σε κάποιο ήδη υπάρχων παιχνίδι που έχει φτιαχτεί από κάποιον άλλο παίκτη αν και εφόσον αυτό δεν έχει ήδη αρχίσει. Όταν επιλέξει κάποιο από τα παιχνίδια της λίστας και πατήσει το κουμπί Join τότε αν δεν έχει παίξει κάποιο άλλο παιχνίδι προηγούμενος ο χρήστης αυτός, του εμφανίζεται μια φόρμα για την συμπλήρωση του ονόματος του για να μπορεί να παίξει, θα του ζητηθεί επίσης στην ίδια φόρμα και κωδικός αν το παιχνίδι είναι ιδιωτικό για να μπορέσει να συμμετέχει σε αυτό. Αν τώρα δεν είναι ούτε ιδιωτικό και ο χρήστης έχει ξαναπαίξει προηγούμενος κάποιο άλλο παιχνίδι, οπότε υπάρχει στην προσωρινή μνήμη το όνομα του, τότε πατώντας το Join απλά μεταφέρεται στην σελίδα του παιχνιδιού χωρίς να συμπληρώσει κάποιο άλλο στοιχείο.
- Στην συνέχεια βρισκόμαστε στην σελίδα του παιχνιδιού η οποία είναι η πιο σημαντική και παράλληλα αυτή που περιέχει την περισσότερη πληροφορία. Στην σελίδα αυτήν βλέπουμε στην αριστερή πλευρά τις πληροφορίες των παικτών, όπου αποτελούνται από το εικονίδιο του χρήστη (avatar), το όνομα του και την βαθμολογία του. Ο

χρήστης του οποίο είναι η σειρά να παίξει είναι πάντα πιο έντονος χρωματικά από του υπόλοιπους. Στο κέντρο της σελίδας υπάρχει το ταμπλό του παιχνιδιού και ακριβώς κάτω από αυτό υπάρχει το ζάρι το οποίο είναι ενεργό μόνο στον χρήστη του οποίου είναι η σειρά να παίξει. Στα δεξιά της οθόνης μας υπάρχουν οι κάρτες των ερωτήσεων και οι κάρτες ενεργειών. Το παιχνίδι αρχίζει να εξελίσσεται αφού ολοκληρωθεί ο χρόνος που είναι διαθέσιμος για να εισέλθουν όλοι οι χρήστες και πατήσει ο πρώτος χρήστης το ζάρι. Στην κάτω δεξιά γωνία υπάρχει ένα κουμπί με τίτλο Exit το οποίο μπορεί ο χρήστης να το χρησιμοποιήσει οποιαδήποτε στιγμή για να αποχωρήσει από το παιχνίδι χωρίς να επηρεάσει την ροή αυτού για τους υπόλοιπους παίκτες.

- Τέλος το παιχνίδι ολοκληρώνεται μόνο όταν όλοι οι χρήστες φτάσουν στο σημείο του τερματισμού όπου εκεί εμφανίζεται ένα αναδυόμενο παράθυρο το οποίο αναφέρει τον νικητή και την βαθμολογία του. Εάν κάποιος φτάσει στον τερματισμό πριν τερματίσουν οι υπόλοιποι παίκτες τότε εμφανίζεται ένα αναδυόμενο παράθυρο και τον ενημερώνει για τον τερματισμό του και την βαθμολογία του, ακόμη τον ενημερώνει ότι θα χρειαστεί να περιμένει να ολοκληρώσουν όλοι οι χρήστες για την ανάδειξη του νικητή.

Σχεδιασμός

Η καταγραφή των απαιτήσεων-προδιαγραφών είναι ίσως η βασικότερη και από τις πιο σημαντικές διαδικασίες πριν την έναρξη της ανάλυσης, του σχεδιασμού και της υλοποίησης ενός έργου. Αν δεν υπάρχουν σωστές προδιαγραφές για κάτι το οποίο θέλουμε να υλοποιήσουμε τότε είναι πολύ εύκολο να φτάσουμε στην δημιουργία ενός αποτελέσματος το οποίο παρεκκλίνει κατά πολύ από το τελικό αποτέλεσμα στο οποίο στόχευε ο πελάτης, με αυτό να συνεπάγεται άστοχη κατανάλωση ανθρώπινων πόρων, χρόνου και χρήματος.
Συνοψίζοντας, όσο πιο καλά, στοχευμένα, αποτελεσματικά και λεπτομερώς γίνεται η καταγραφή των απαιτήσεων, η ανάλυση αυτών και η εξαγωγή των προδιαγραφών, τόσο πιο γρήγορα, εύκολα και ικανοποιητικά υλοποιείται ένα έργο.

Καταγραφή και Ανάλυση απαιτήσεων

Το πρώτο στοιχείο που θα πρέπει να έχει γίνει πλήρως ξεκάθαρο σε αυτόν ή αυτούς οι οποίοι θα αναλάβουν την υλοποίηση ενός έργου, πριν ξεκινήσει η παραγωγή του, είναι οι προδιαγραφές και ο σκοπός του έργου το οποίο θέλουμε να δημιουργήσουμε. Σε πολύ γενικές γραμμές μας ζητήθηκε να δημιουργήσουμε ένα διαδικτυακό παιχνίδι το οποίο θα πρέπει να έχει πρωτότυπο σενάριο αλλά ταυτόχρονα μας δόθηκε η δυνατότητα να εμπεριέχει στοιχεία από άλλα ήδη υπάρχοντα παιχνίδια.
Βασιζόμενοι λοιπόν στα προηγούμενα και επειδή οι απαιτήσεις του παιχνιδιού ήταν δική μας ευθύνη να τις καθορίσουμε, αποφασίσαμε ότι το κοινό στο οποίο θα απευθυνθεί το παιχνίδι μας ανήκει σε ηλικίες μεγαλύτερες από αυτές τις οποίες απευθύνεται το παιχνίδι στο οποίο βασιστήκαμε. Πιο συγκεκριμένα το παιχνίδι μας θα απευθύνεται σε ηλικίες 15+ κυρίως λόγω των ερωτήσεων γνώσεων που εμπεριέχει και έτσι το κοινό μας θα πρέπει να έχει κάποιες βασικές γνώσεις πληροφορικής. Το συγκεκριμένο πεδίο επιλέχτηκε πιλοτικά, βάση της ανάγκης να περιορίσουμε το εύρος και τον όγκο των απαραίτητων ερωτήσεων για την δημιουργία ενός παιχνιδιού για τις ανάγκες της εργασίας μας. Παρόλα αυτά το παιγνίδι είναι σχεδιασμένο με τρόπο τέτοιο που επιτρέπει την εύκολη προσθήκη άλλων θεματικών περιοχών ερωτήσεων.
Επίσης το παιχνίδι δεν θέλαμε να είναι υψηλής δυσκολίας ώστε να είναι εύκολα προσεγγίσιμο από ανθρώπους οι οποίοι δεν θέλουν να δυσκολευτούν με την κατανόηση του για να συμμετέχουν σε αυτό με απώτερο σκοπό την δημιουργία αισθήματος χαράς και διασκέδασης αλλά και παράλληλα την δημιουργία στρατηγικής σκέψης και απόκτησης γνώσεων, κυρίως λόγω της χρήσης των καρτών ενεργειών στις οποίες αναφερθήκαμε και εξηγήσαμε λεπτομερώς στο προηγούμενο κεφάλαιο αλλά και των ερωτήσεων που εμπεριέχει, με τελικό σκοπό το επιθυμητό αποτέλεσμα που δεν είναι άλλο από την επίτευξη νίκης του παιχνιδιού.
Συνεχίζοντας την ανάλυση των προδιαγραφών, το παιχνίδι θελήσαμε να έχει σκιτσογραφική μορφή, χρησιμοποιώντας εικόνες για τους παίκτες που παραπέμπουν σε μορφές φιδιών με παιδικό ύφος. Επίσης τα χρώματα που χρησιμοποιήθηκαν αλλά και οι τόνοι αυτών όπως θα

δούμε και στις εικόνες παρακάτω είναι έντονοι με σκοπό να τραβούν στην προσοχή του εκάστοτε παίκτη και να δημιουργούν την αίσθηση του παιχνιδιού στο οποίο βασιστήκαμε, το οποίο στις εκδόσεις που κυκλοφορεί στις μέρες μας έχει αντιστοιχών εντάσεων χρωματισμούς.
Επίσης θέλαμε να μπορούν να συμμετέχουν από δύο μέχρι τέσσερις παίκτες και να δίνεται η δυνατότητα να μπορούν να εκτελούνται ταυτόχρονα παραπάνω από ένα στιγμιότυπα του παιχνιδιού. Δηλαδή να μπορούν να παίζονται παράλληλα πολλά παιχνίδια, με τον μόνο περιορισμό για τον αριθμό αυτών, να είναι τεχνικός και έχει να κάνει με τα χαρακτηριστικά του διακομιστή (server) στον οποίο είναι εγκατεστημένη η εφαρμογή μας.

Σχεδιασμός Οθονών και Προτύπων

Τεχνολογίες

Οι τεχνολογίες και τα προγράμματα που χρησιμοποιήθηκαν για την υλοποίηση του γραφιστικού τμήματος του παιχνιδιού αλλά και του σχεδιαστικού των οθόνων είναι οι εξής παρακάτω:

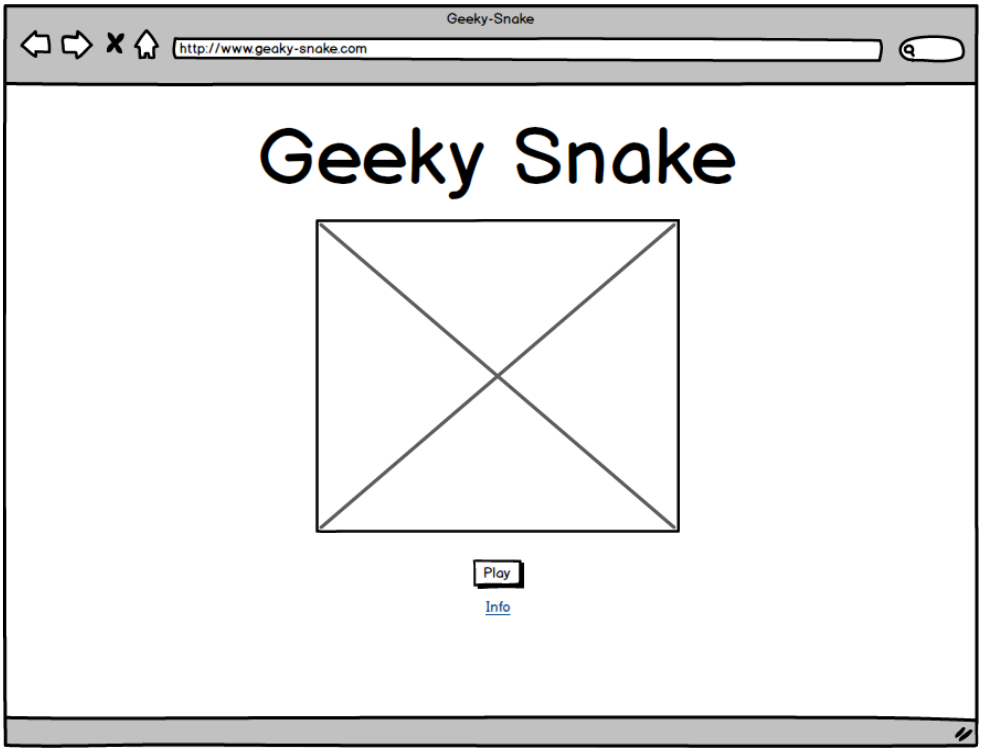
- Για τον σχεδιασμό του ταμπλό, των εικόνων των χρηστών, των καρτών, του λογοτύπου αλλά και του ονόματος του παιχνιδιού χρησιμοποιήθηκαν τα προγράμματα της εταιρίας Adobe Photoshop και Illustrator. (Adobe Photoshop) (Abode Illustrator)
- Για τον σχεδιασμό των προσχέδιων οθόνων χρησιμοποιήθηκε η διαδικτυακή πλατφόρμα https://moqups.com/ και το πρόγραμμα balsamiq. (Balsamiq)
- Για το σχεδιασμό της ιστοσελίδας που περιέχει το παιχνίδι μας χρησιμοποιήθηκε η τεχνολογία CSS3 βασισμένη στο framework Bootstrap. Στην συνέχεια θα αναφερθούμε λεπτομερώς στις τεχνολογίες αυτές.

Παρουσίαση των προτύπων και οθονών

Στην ενότητα αυτή θα παρουσιάσουμε τα προσχέδια των οθόνων (mockups) που δημιουργήσαμε μαζί με στιγμιότυπα από τις υπάρχουσες οθόνες του παιχνιδιού μας και τα γραφικά στα οποία αναφερθήκαμε προηγουμένως. Επίσης θα δούμε και κάποια πρότυπα, τα οποία απορρίψαμε από την τελική επιλογή μας, για τα οποία θα αναφερθεί και ο λόγος απόρριψης.

Αρχική Οθόνη

Θα ξεκινήσουμε παρουσιάζοντας την αρχική οθόνη μας και το πρότυπο που έχουμε σχεδιάσει γι αυτήν. Στην *εικόνα 8* έχουμε το προσχέδιο της αρχικής οθόνης του παιχνιδιού μας, την οποία βλέπουμε όταν πληκτρολογούμε την διεύθυνση www.geeky-snake.com και στην *εικόνα 9* βλέπουμε την αρχική εικόνα όπως είναι τώρα στο παιχνίδι μας. Παρατηρούμε ότι σχεδιαστικά είναι ακριβώς όπως την είχαμε δημιουργήσει και στο προσχέδιο χωρίς ιδιαίτερες αλλαγές. Στο πάνω μέρος της οθόνης μας υπάρχει το όνομα του παιχνιδιού "Geeky Snake", ακριβώς από κάτω βρίσκεται το λογότυπο μας, και κάτω από αυτό βρίσκεται το λειτουργικό τμήμα της οθόνη όπου αποτελείτε από δύο κουμπιά. Το κουμπί με όνομα "PLAY" του οποίου η λειτουργία είναι να μας μεταφέρει στην σελίδα δημιουργίας νέου ή επιλογής από τα ήδη υπάρχοντα παιχνίδια και το δεύτερο κουμπί με τίτλο "info" μας μεταφέρει στην σελίδα στην οποία περιγράφονται οι κανόνες του παιχνιδιού.

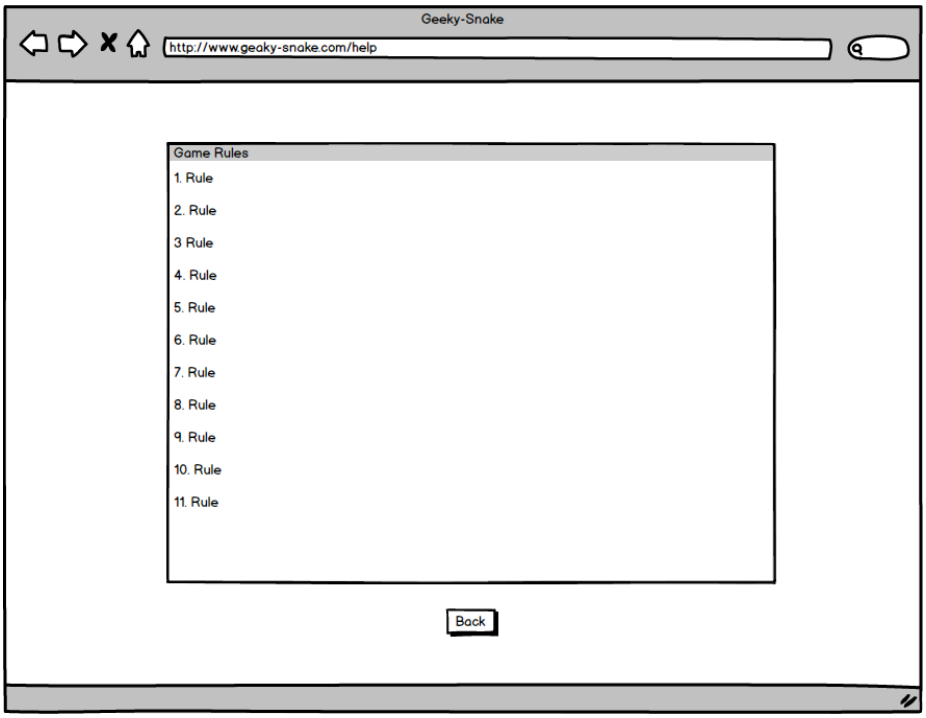


Εικόνα 8. Πρώτο προσχέδιο αρχικής σελίδας

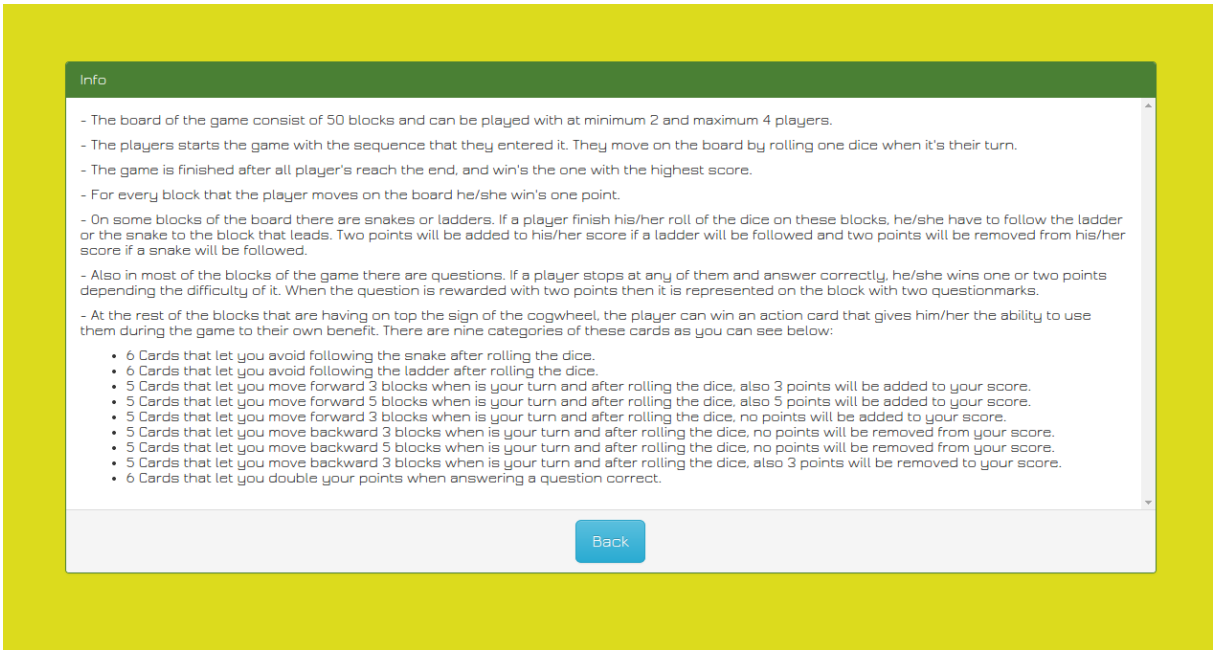
Εικόνα 9. Υλοποίηση προσχεδίου αρχικής σελίδας

Οθόνη Περιγραφής Κανόνων

Συνεχίζοντας στην οθόνη που περιγράφουμε και παρουσιάζουμε τους κανόνες του παιχνιδιού βλέπουμε ότι και εδώ ο αρχικός σχεδιασμός που βλέπουμε στην *εικόνα 10* είναι ο ίδιος με τον τρέχον του παιχνιδιού μας που έχουμε στην *εικόνα 11*. Επίσης παρατηρούμε ότι υπάρχει ένα κουμπί με τίτλο "Back" του οποίου η λειτουργία είναι να μας μεταφέρει στην αρχική οθόνη, δηλαδή πίσω σε αυτήν που αναφέραμε ακριβώς από πάνω.



Εικόνα 10. Σελίδα κανόνων παιγνιδιού

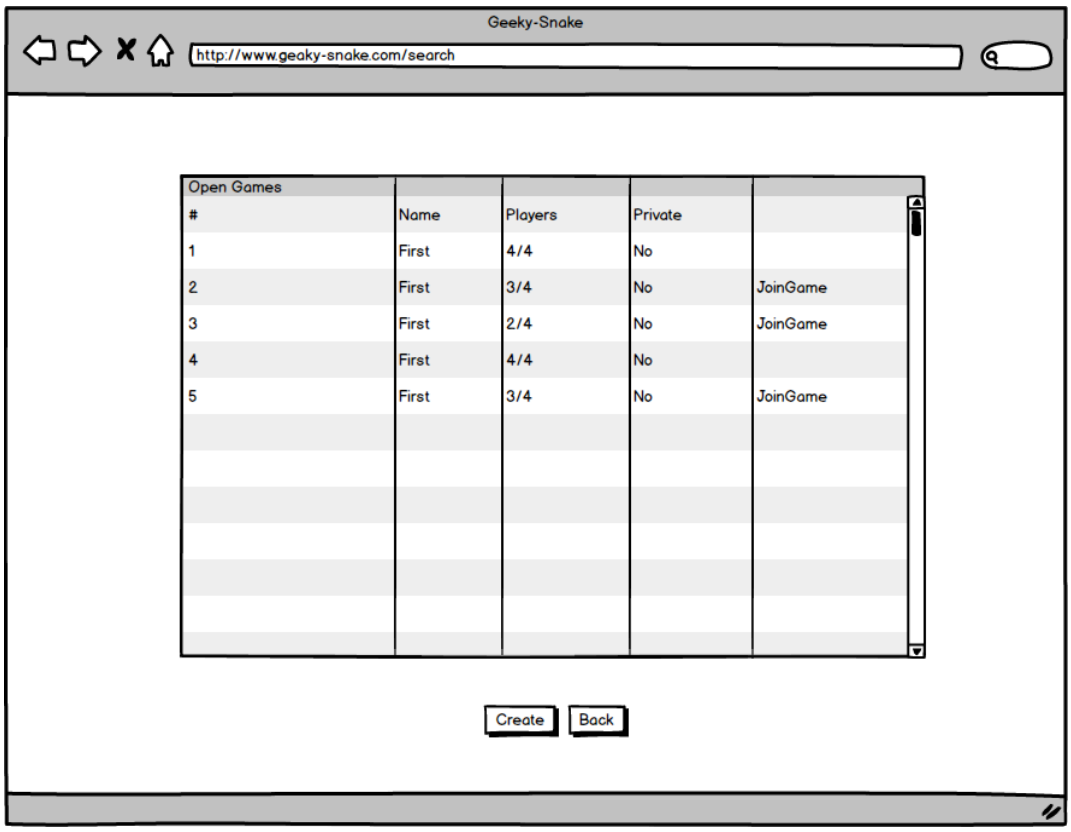


Εικόνα 11. Υλοποίηση σελίδας πληροφοριών

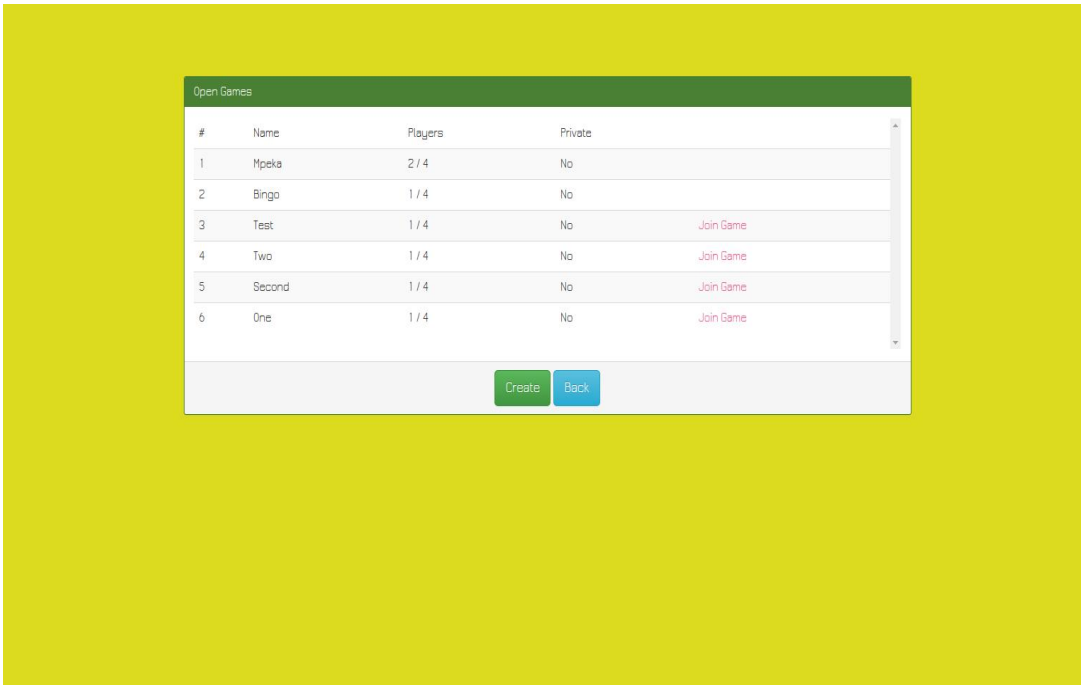
Οθόνη Δημιουργίας – Αναζήτησης Παιχνιδιού

Επόμενη οθόνη που θα εξετάσουμε είναι αυτή της δημιουργίας ή της επιλογής κάποιου παιχνιδιού. Και αυτή η οθόνη όπως παρατηρούμε στην *εικόνα 12* έχει παραμείνει ίδια ακριβώς με το αρχικό προσχέδιο που βλέπουμε στην *εικόνα 13*. Παρατηρούμε ότι υπάρχει στο μέσο της οθόνης μας ένας πίνακας ο οποίος περιέχει όλα τα παιχνίδια που παίζονται αυτήν την στιγμή είτε αυτά είναι διαθέσιμα να συμμετέχουμε είτε όχι για συμμετοχή. Στην συνέχεια θα εξηγήσουμε τι σημαίνει διαθέσιμο προς συμμετοχή παιχνίδι. Στο πάνω μέρος του πίνακα υπάρχει το όνομα "Open Games", κάτω από αυτό υπάρχουν τα ονόματα που περιγράφουν τα περιεχόμενα κάθε στήλης του πίνακα. Στην πρώτη στήλη υπάρχει ένας αύξων αριθμός ο οποίος αριθμεί τα υπάρχοντα παιχνίδια και ακριβώς δίπλα υπάρχει το όνομα του κάθε παιχνιδιού. Δίπλα από το όνομα αναφέρονται πόσοι, από το μέγιστο που είναι τέσσερις, παίκτες έχουν συνδεθεί ή συμμετέχουν στο συγκεκριμένο στιγμιότυπο του παιχνιδιού. Ακριβώς μετά από τον αριθμό των παικτών βρίσκεται η πληροφορία του αν το παιχνίδι είναι δημόσιο ή όχι, δηλαδή αν για να συνδεθεί κάποιος σε αυτό απαιτείται κωδικός, ο οποίος καθορίζεται όπως θα δούμε και παρακάτω στην φόρμα δημιουργίας του παιχνιδιού. Στην τελευταία στήλη του πίνακα υπάρχει ένας σύνδεσμος με τίτλο "Join Game" ο οποίος μας επιτρέπει να συνδεθούμε σε ένα παιχνίδι αν αυτό είναι διαθέσιμο και δεν έχει ήδη ξεκινήσει ή έχει συμπληρώσει τέσσερις παίκτες. Όπως παρατηρούμε στο δεξί μέρος του πίνακα υπάρχει μια κατακόρυφη μπάρα κύλισης, αυτό σημαίνει ότι αν τα τρέχοντα στιγμιότυπα του παιχνιδιού ξεπεράσουν το μέγεθος του πίνακα θα μπορούμε να δούμε αυτά που δεν χωράνε στο ορατό κομμάτι του πίνακα απλά κυλώντας την μπάρα. Με αυτό πετυχαίνουμε να έχουμε πάντα σε πρώτη θέαση τα δύο κουμπιά που βρίσκονται κάτω από τον πίνακα για τα οποία θα εξηγήσουμε την λειτουργία τους παρακάτω. Η σειρά με την οποία εμφανίζονται τα παιχνίδια

είναι χρονολογική από το πιο νέο στο πιο παλαιό. Επίσης όταν βρισκόμαστε στην σελίδα δημιουργίας – αναζήτησης παιχνιδιού και κάποιος άλλος χρήστης δημιουργήσει ένα νέο παιχνίδι τότε αυτόματος ενημερώνεται και γίνεται διαθέσιμο το παιχνίδι αυτό και στη λίστα του πίνακά που βλέπουν οι υπόλοιποι χρήστες χωρίς να χρειάζεται να γίνει ανανέωση της σελίδας μας.

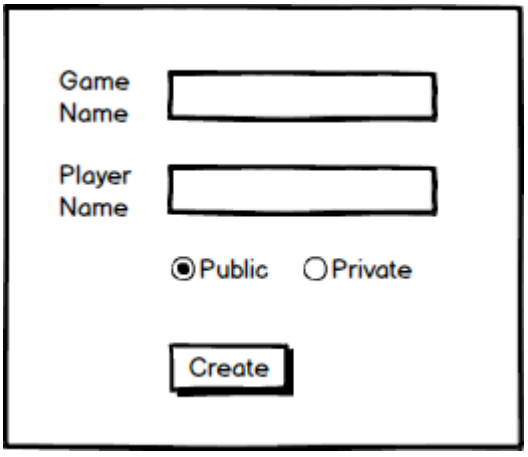


Εικόνα 12. Προσχέδιο σελίδας διαθέσιμων παιγνιδιών



Εικόνα 13. Υλοποίηση σελίδας παιγνιδιών

Η λειτουργία του κουμπιού "Back" είναι να μας μεταφέρει πίσω στην αρχική σελίδα του παιχνιδιού μας την οποία είδαμε παραπάνω στην *εικόνα 9*. Η λειτουργία του κουμπιού "Create" είναι να εμφανίζει μία φόρμα συμπλήρωσης στοιχείων, της οποίας το προσχέδιο βλέπουμε στην εικόνα 14, τα οποία απαιτούνται για την δημιουργία ενός νέου στιγμιότυπου του παιχνιδιού μας. Στην εικόνα 15 βλέπουμε πως είναι η τελική μορφή της φόρμας μες το παιχνίδι μας. Ο χρήστης μπορεί να κλείσει την φόρμα απλός κάνοντας κλικ με το ποντίκι του σε κάποιο μέρος της οθόνης που είναι εκτός της φόρμας.

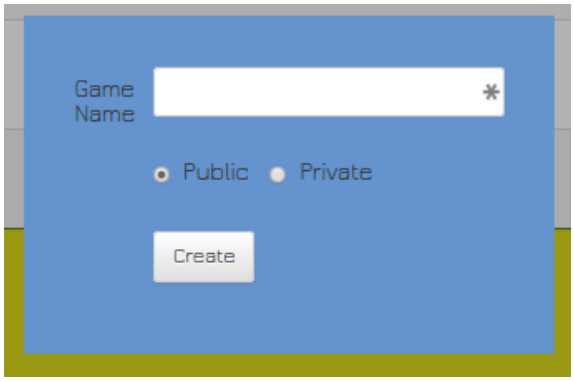


Εικόνα 14. Προσχέδιο σελίδας δημιουργίας παιγνιδιού

Εικόνα 15. Υλοποίηση σελίδας δημιουργίας παιγνιδιού

Στην φόρμα αυτή μας ζητάτε το όνομα που θέλουμε να δώσουμε στο στιγμιότυπο του παιχνιδιού το οποίο θα δημιουργήσουμε, το όνομα του παίκτη που θα είναι ο δημιουργός και παράλληλα θα συμμετέχει και στο παιχνίδι, επίσης μας δίνεται η επιλογή αν το παιχνίδι που δημιουργούμε θα είναι ιδιωτικό ή προσβάσιμο από όλους τους χρήστες, τέλος πατώντας το κουμπί "Create" θα δημιουργείται το παιχνίδι μας και θα αποθηκεύεται στην βάση δεδομένων. Στην συνέχεια ο χρήστης με την ολοκλήρωση θα μεταφέρεται στην σελίδα του παιχνιδιού.

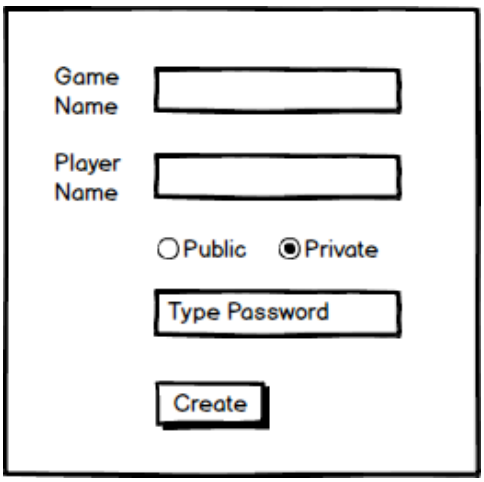
Συνεχίζοντας αν ο χρήστης έχει δημιουργήσει και κάποιο άλλο παιχνίδι προηγουμένως ή έχει συμμετάσχει στο παιχνίδι κάποιου άλλου χρήστη και έχει ήδη δώσει όνομα για την δημιουργία κάποιου χρήστη, τότε σε αυτόν αν θέλει να δημιουργήσει ένα νέο παιχνίδι δεν θα του ζητάτε όνομα στην φόρμα συμπλήρωσης, όπως φαίνεται και στην *εικόνα 16,* παρά μόνον το όνομα του παιχνιδιού και αν είναι ιδιωτικό ή όχι.



Εικόνα 16. Δημιουργία παιγνιδιού για συνδεδεμένο χρήστη

Στην *εικόνα 17* βλέπουμε το προσχέδιο της φόρμας συμπλήρωσης στοιχείων όταν κάποιος χρήστης θελήσει να δημιουργήσει ένα νέο παιχνίδι με επιλεγμένη την επιλογή "Private". Στο σημείο αυτό υπάρχει ένα ακόμη πεδίο προς συμπλήρωση το οποίο είναι αυτό του κωδικού.

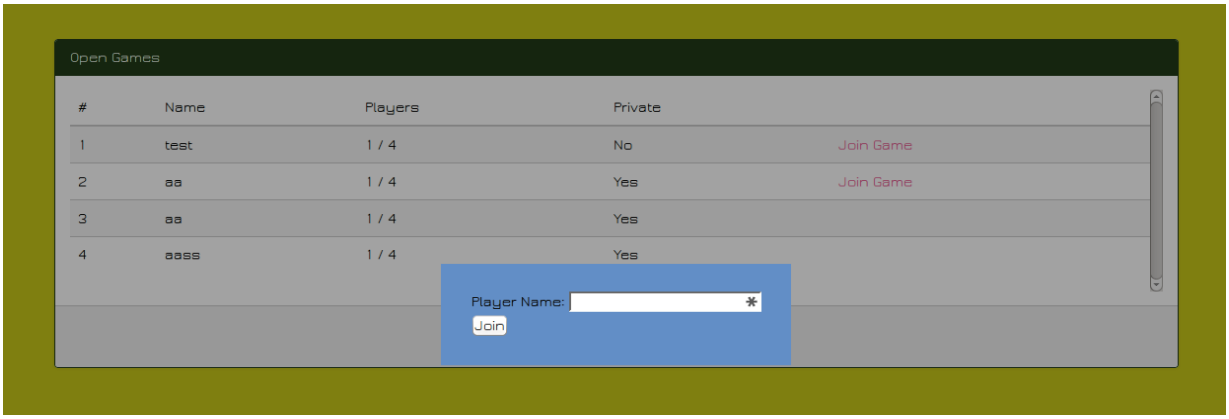
Στο παιχνίδι αυτό θα μπορεί να συμμετέχει μόνο κάποιος που γνωρίζει τον κωδικό αυτό. Στην *εικόνα 18* βλέπουμε την τελική μορφή της φόρμα που είδαμε στο προσχέδιο της *εικόνας 17*.



Εικόνα 17. Προσχέδιο σελίδας δημιουργίας ιδιωτικού παιγνιδιού

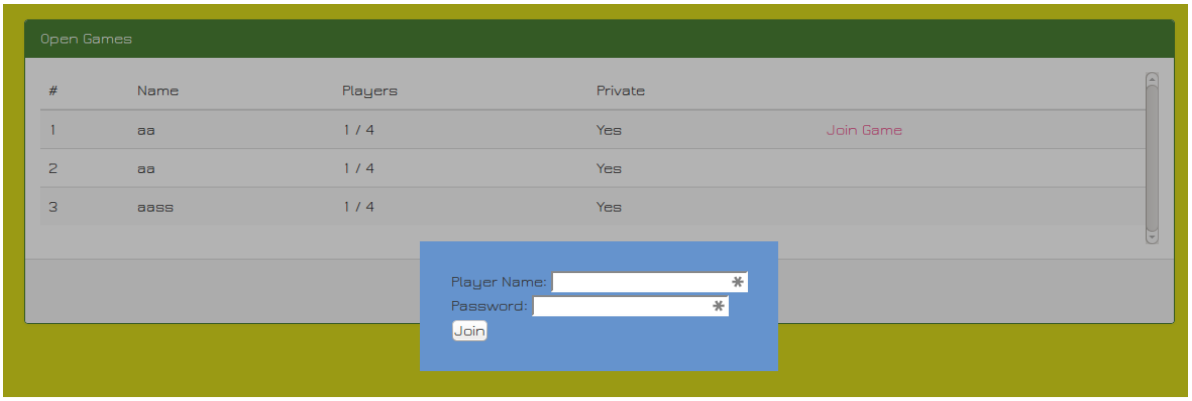
Εικόνα 18. Υλοποίηση σελίδας δημιουργίας ιδιωτικού παιγνιδιού

Ο σύνδεσμος " Join Game" που υπάρχει δίπλα σε κάποια από τα παιχνίδια που υπάρχουν στον πίνακα μας είναι για να μπορεί κάποιος χρήστης να συμμετέχει σε αυτό. Αν αυτός δεν υπάρχει σημαίνει ότι το παιχνίδι έχει ήδη ξεκινήσει και δεν μπορεί πια να συμμετέχει κάποιος σε αυτό. Όταν επιλεχθεί ο σύνδεσμος Join Game και ο χρήστης είναι η πρώτη φορά που παίζει κάποιο παιχνίδι τότε θα εμφανιστεί μια φόρμα στην οποία θα του ζητάτε, όπως βλέπουμε και στην *εικόνα 19*, το όνομα του με το οποίο θα εμφανίζεται μέσα στο παιχνίδι.



Εικόνα 19. Δήλωση ονόματος παίκτη

Αν το παιχνίδι είναι ιδιωτικό και ο χρήστης είναι πρώτη φορά που παίζει, εκτός από το όνομα θα του ζητηθεί να εισάγει και ένα κωδικό, όπως βλέπουμε και στην εικόνα 20, ο οποίος είναι αυτός που είχε θέσει ο δημιουργός αυτού.



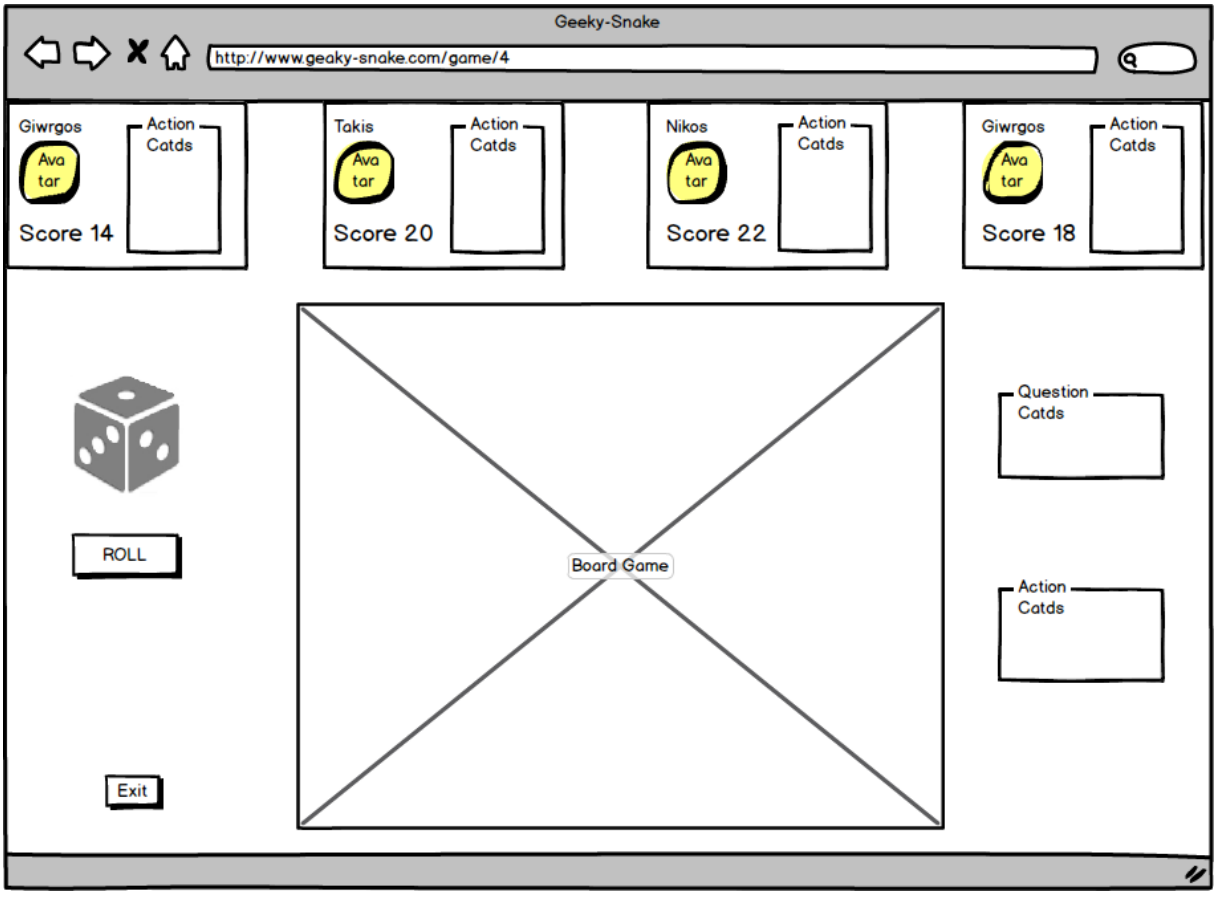
Εικόνα 20. Σύνδεση σε ιδιωτικό παιγνίδι

Τέλος αν ένας χρήστης που θέλει να συμμετέχει σε ένα παιχνίδι, επιλέγοντας τον σύνδεσμο "Join Game", έχει ξαναπαίξει προηγουμένως κάποιο παιχνίδι και υπάρχει ακόμη στην προσωρινή μνήμη το όνομά του και επίσης αυτό το παιχνίδι που θέλει να συμμετέχει δεν είναι ιδιωτικό, τότε πατώντας τον σύνδεσμο "Join Game" χωρίς να του ζητηθεί καμία πληροφορία μεταφέρεται στην σελίδα του παιχνιδιού και εμφανίζεται με το όνομα που είχε εισάγει στο πρώτο παιχνίδι που συμμετείχε.

Οθόνη Παιχνιδιού

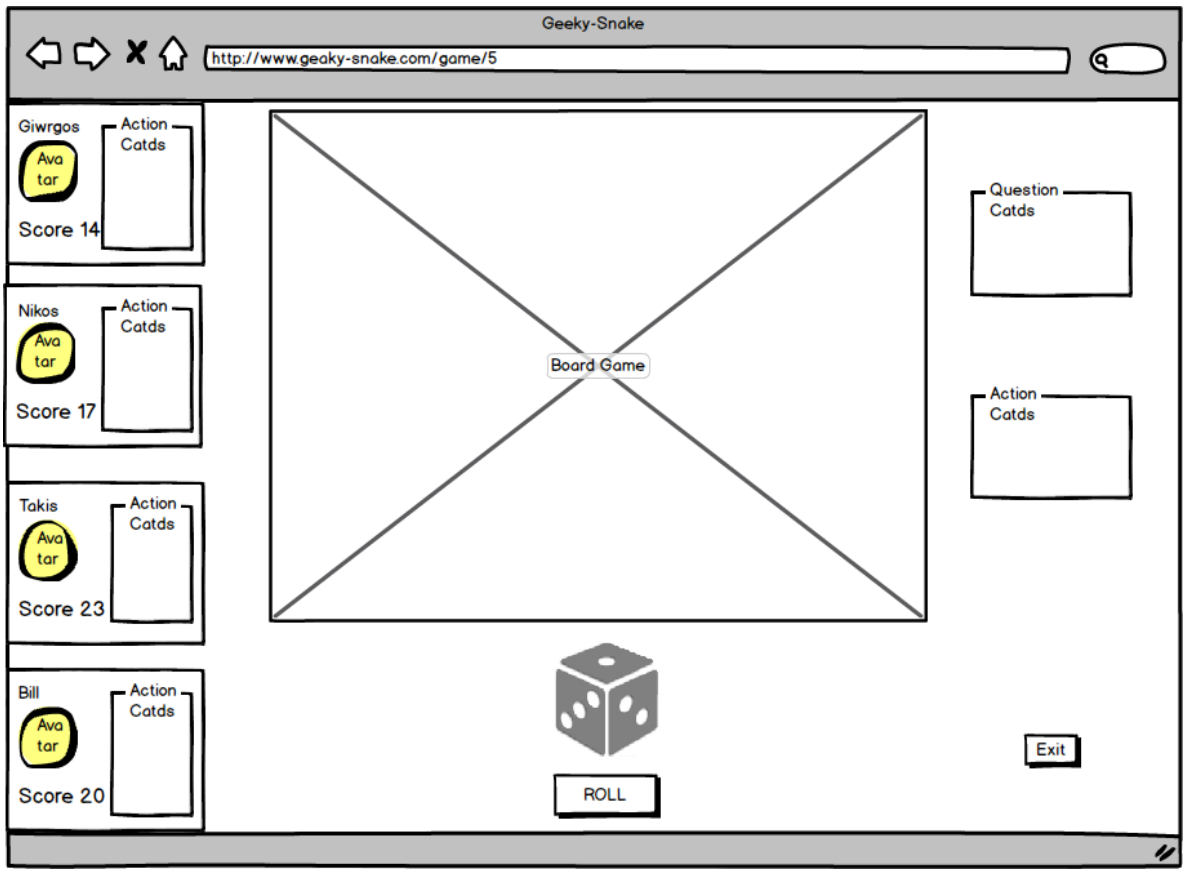
Τελευταία οθόνη του παιχνιδιού μας και πιο σημαντική είναι αυτή της ανάπτυξης του παιχνιδιού. Στην οθόνη μας αυτή θα παρουσιάσουμε τα δύο επικρατέστερα προσχέδια που είχαμε φτιάξει πριν ξεκινήσει η υλοποίηση του παιχνιδιού και θα εξηγήσουμε πως πάρθηκε η επιλογή μέσα από αυτά τα δύο.
Στις *εικόνες 21, 22* βλέπουμε τα δύο προσχέδια. Αυτό που τελικά επικράτησε είναι αυτό της *εικόνας 22* με τους λόγους να είναι και τεχνικοί αλλά και παρουσίασης. Στην εικόνα 21

παρατηρούμε στο πάνω μέρος της οθόνης μας να είναι τοποθετημένες οι πληροφορίες όλων των παικτών. Στις πληροφορίες των παικτών περιέχονται το όνομα, η εικόνα (avatar) του κάθε παίκτη η οποία είναι αυτή με την οποία αντιστοιχίζεται και κινείτε ο παίκτης πάνω στο ταμπλό, όπως επίσης αναφέρεται το τρέχον σκορ και οι κερδισμένες "Action Cards" αυτού. Στην αριστερή πλευρά του προσχεδίου βρίσκεται το ζάρι του παιχνιδιού και κάτω από αυτό υπάρχει ένα κουμπί με τίτλο "Exit" του οποίου η λειτουργία είναι να μας μεταφέρει στην σελίδα αναζήτησης των παιχνιδιών που αναφέραμε παραπάνω. Συνεχίζοντας στην δεξιά πλευρά παρατηρούμε τοποθετημένες τις κάρτες των ερωτήσεων αλλά και των ενεργειών του παιχνιδιού και τέλος στο κεντρικό και κάτω μέρος της οθόνης μας υπάρχει το ταμπλό του παιχνιδιού όπου πάνω σε αυτό εκτυλίσσεται το παιχνίδι.



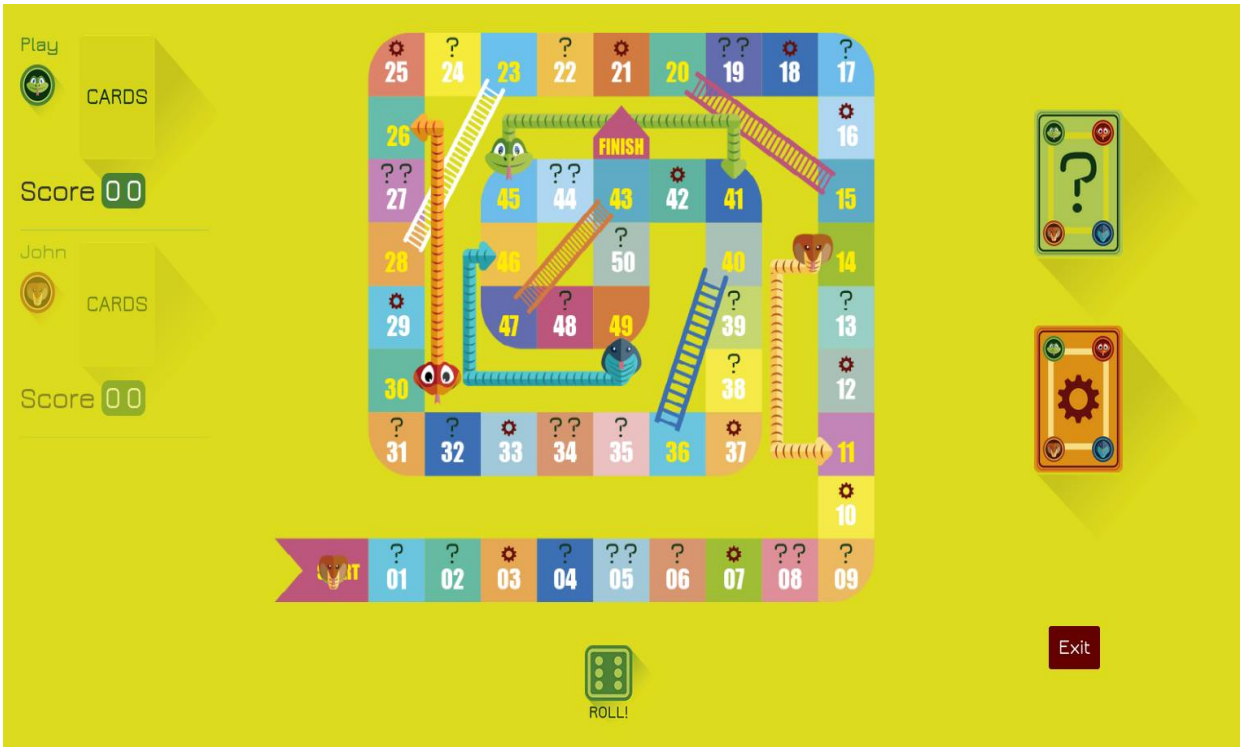
Εικόνα 21. Προσχέδιο σελίδας εκτέλεσης παιγνιδιού

Στο προσχέδιο της *εικόνας 22* βλέπουμε ότι οι πληροφορίες των χρηστών βρίσκονται στην αριστερή πλευρά της οθόνης μας. Στην δεξιά υπάρχουν και πάλι οι κάρτες ερωτήσεων και ενεργειών μαζί με το κουμπί εξόδου από το παιχνίδι. Στο κάτω μέρος της οθόνης μας βρίσκεται το ζάρι και τέλος το ταμπλό του παιχνιδιού βρίσκεται στο πάνω και κεντρικό σημείο της οθόνης μας.



Εικόνα 22. Εναλλακτικό προσχέδιο σελίδας εκτέλεσης παιγνιδιού

Οι κύριοι λόγοι για την επιλογή του δεύτερου προσχέδιου της *εικόνας 22* έγινε γιατί στο επίκεντρο της προσοχής του παίκτη θέλαμε να είναι το ταμπλό του παιχνιδιού και από την αξιολόγηση των προσχεδίων που κάναμε μέσω ερωτήσεων σε πιθανούς χρήστες του παιχνιδιού κατανοήσαμε ότι στην αρχική εκδοχή το πρώτο αντικείμενο το οποίο παρατηρούσε ο υποψήφιος παίκτης ήταν οι πληροφορίες των χρηστών που βρίσκονται στο πάνω μέρος της οθόνης. Επίσης υπάρχει και δευτερεύοντες τεχνικοί λόγοι καθώς στην περίπτωση που υπήρχαν λιγότεροι των τεσσάρων παικτών εμφανιζόταν κενός χώρος στο σημείο της οθόνης όπου εμφανίζονται οι πληροφορίες των παικτών ο οποίος δεν ήταν ωραίος αισθητικά (διαχώριζε πληροφορίες που έπρεπε να βρίσκονται σε συνέχεια). Παίρνοντας υπόψη όλα τα προηγούμενα καταλήξαμε στην εκδοχή της *εικόνας 22* την οποία παρατηρούμε πώς ακριβώς είναι η τελική της μορφή στην *εικόνα 23*.

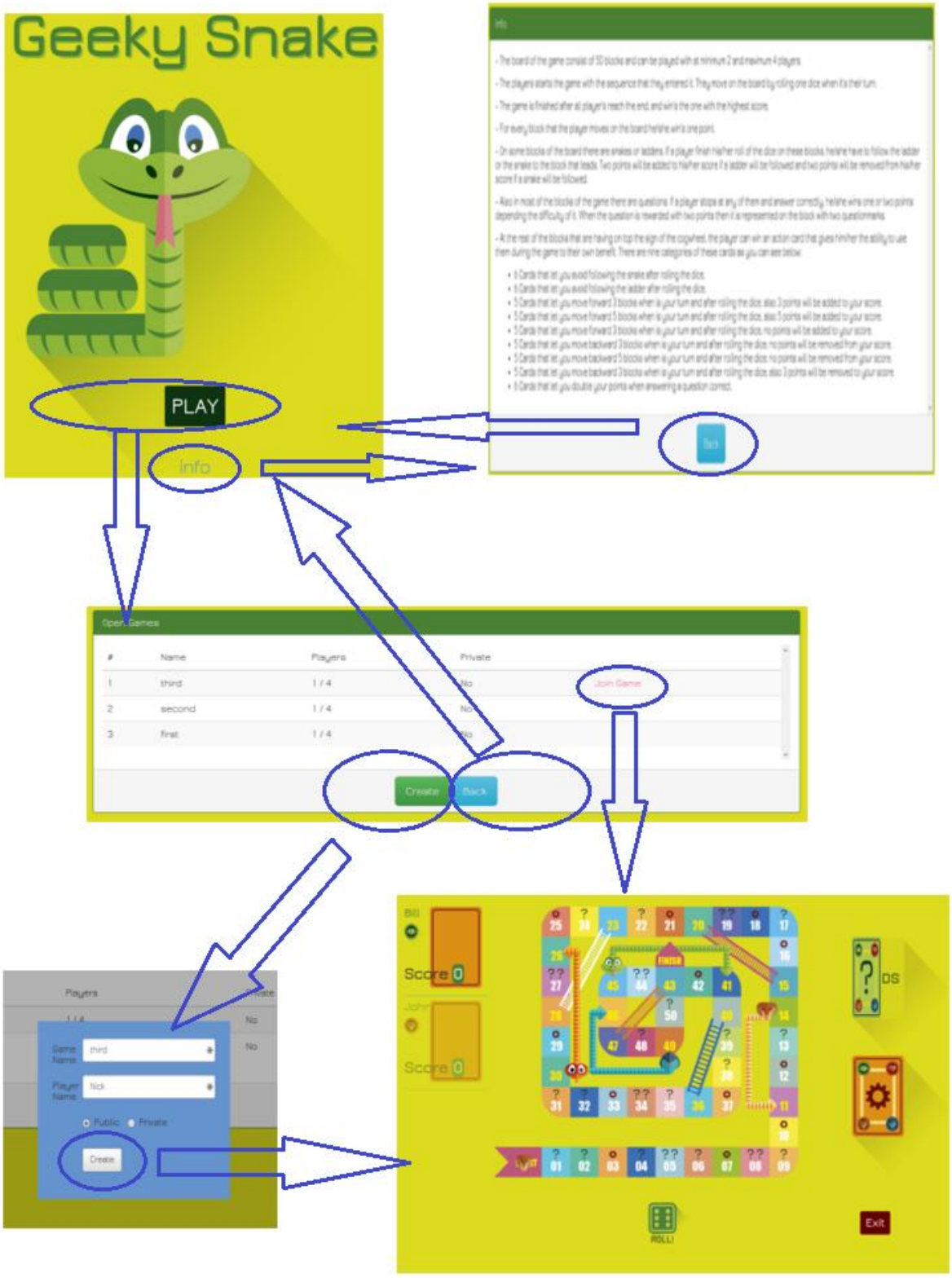


Εικόνα 23. Υλοποίηση σελίδας εκτέλεσης παιγνιδιού

Στην *εικόνα 24* παρατηρούμε τις μορφές (avatars) των παικτών του παιχνιδιού. Κάθε χρήστης που θα συμμετέχει σε ένα παιχνίδι θα αντιπροσωπεύεται σε αυτό με μία από τις παρακάτω μορφές, οι οποίες θα υπάρχουν στην αριστερή πλευρά της οθόνης μας που υπάρχουν οι πληροφορίες του κάθε χρήστη όπως παρατηρούμε στη *εικόνα 23*, αλλά θα είναι και οι ίδιες που θα υπάρχουν πάνω στο ταμπλό και θα κινούνται σε αυτό όταν ο παίκτης μας χρησιμοποιεί το ζάρι.

Εικόνα 24. Εικόνες παικτών

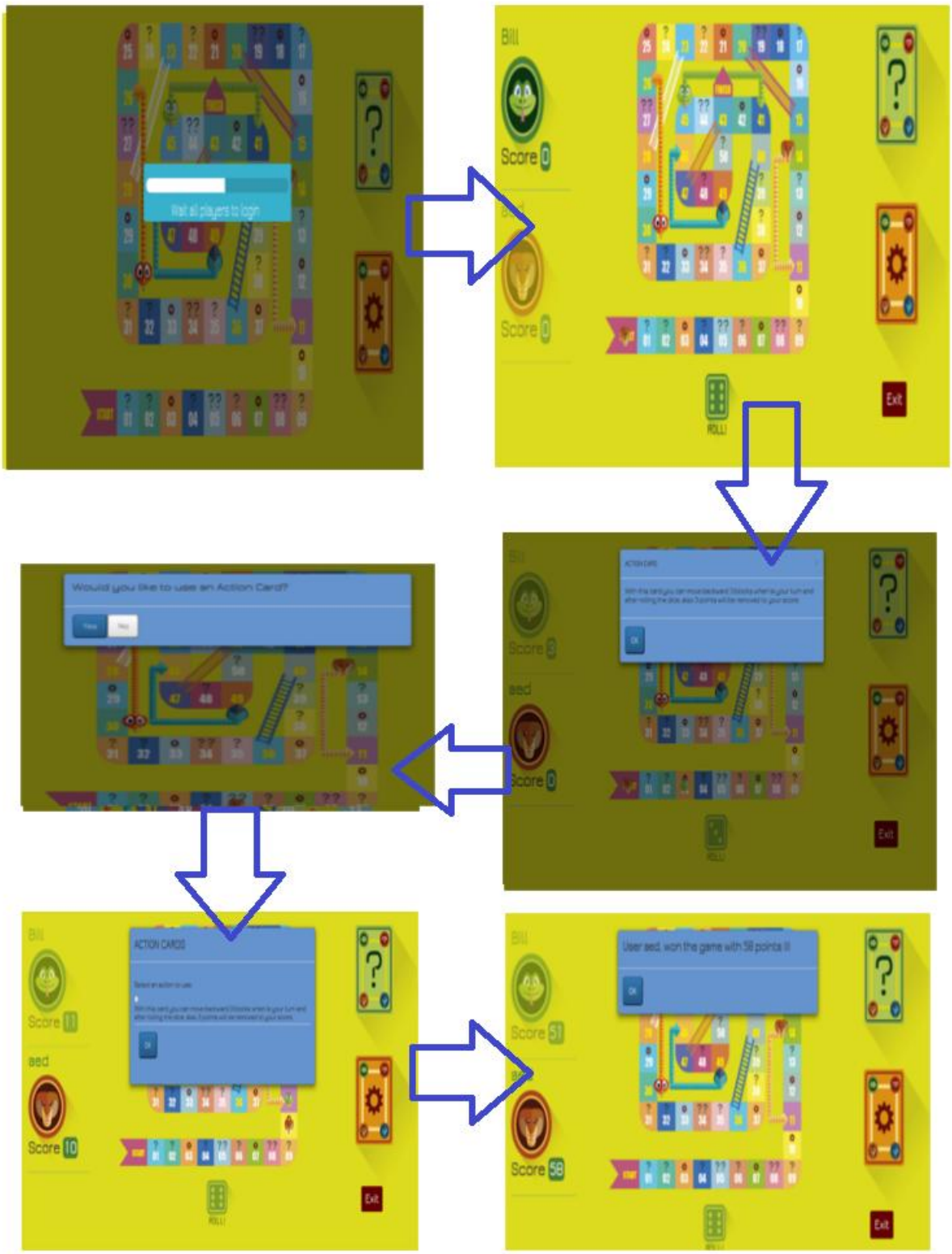
Διάγραμμα ροής δημιουργίας παιχνιδιού



Εικόνα 25 Διάγραμμα ροής δημιουργίας παιχνιδιού

Στην εικόνα 25 παρατηρούμε περιγραφικά μέσω ενός διαγράμματος ροής την διαδικασία η οποία θα πρέπει να ακολουθηθεί για την δημιουργία ενός στιγμιότυπου παιχνιδιού μέσω εικόνων αποτυπωμένων από τα παιχνίδι μας.
Με την πληκτρολόγηση της ηλεκτρονικής διεύθυνσης του παιχνιδιού μας η πρώτη εικόνα που εμφανίζεται είναι η πάνω αριστερά στο διάγραμμα μας η οποία περιέχει δύο κουμπιά. Αν επιλέξουμε το κουμπί με τίτλο Info μεταβαίνουμε στην πάνω δεξιά εικόνα ή οποία είναι η σελίδα που παρουσιάζει του κανόνες του παιχνιδιού μας. Για να μεταβούμε πίσω στην αρχική μας σελίδα από την σελίδα των κανόνων, επιλέγουμε το κουμπί Back. Στην αρχική σελίδα παρατηρούμε επίσης ένα κουμπί το οποίο έχει τίτλο PLAY, αν αυτό επιλεχθεί μας μεταφέρει στην σελίδα δημιουργίας ή επιλογής παιχνιδιού. Σε αυτήν την σελίδα μας δίνεται η δυνατότητα να δούμε όλα τα στιγμιότυπα παιχνιδιών τα οποία παίζονται αυτήν την στιγμή ή έχουν δημιουργηθεί και δεν έχουν ακόμη ξεκινήσει. Στην σελίδα αυτή έχουμε τρείς επιλογές η πρώτη είναι να επιστρέψουμε πίσω στην αρχική μας σελίδα επιλέγοντας το κουμπί με χρώμα μπλε και τίτλο Back, η δεύτερη είναι εάν υπάρχει κάποιο παιχνίδι το οποίο έχει δημιουργηθεί αλλά δεν έχει ξεκινήσει ακόμη τότε δίπλα σε αυτό υπάρχει ο σύνδεσμος με τίτλο Join Game, αν αυτός επιλεχθεί τότε μεταφερόμαστε στην σελίδα του παιχνιδιού την οποία βλέπουμε στην κάτω δεξιά εικόνα του διαγράμματος. Αν δεν υπάρχει κάποιο διαθέσιμο προς συμμετοχή παιχνίδι τότε μπορούμε να δημιουργήσουμε ένα δικό μας, επιλέγοντας το κουμπί με χρώμα πράσινο και όνομα Create το οποίο μας εμφανίζει μια φόρμα συμπλήρωσης στοιχείων για της ανάγκες του παιχνιδιού μας. Μόλις συμπληρωθούν τα απαραίτητα πεδία της φόρμας αυτής και επιλέξουμε το κουμπί Create τότε μεταβαίνουμε στο παιχνίδι το οποίο μόλις δημιουργήσαμε. Αυτοί οι τρόποι είναι και οι μοναδικοί για να μπορέσει κάποιος παίκτης να μεταφερθεί στην σελίδα του εξελίσσεται ένα στιγμιότυπο του παιχνιδιού μας.

Διάγραμμα ροής εξέλιξης στιγμιότυπου παιχνιδιού



Εικόνα 26 Διάγραμμα ροής εξέλιξης παιχνιδιού

Στην *εικόνα 26* παρατηρούμε περιληπτικά την διαδικασία εξέλιξης ενός στιγμιότυπου παιχνιδιού. Στην πρώτη εικόνα πάνω αριστερά βλέπουμε την οθόνη που αντικρίζει ο παίκτης μας καθώς ολοκληρώνει την εισαγωγή του στο στιγμιότυπο του παιχνιδιού. Σε αυτήν όλες οι λειτουργίες του παιχνιδιού είναι απενεργοποιημένες και παρατηρούμε ότι υπάρχει μια κυλιομένη μπάρα η οποία αντιπροσωπεύει τον χρόνο αναμονής μέχρι όλοι οι παίκτες να συνδεθούν σε αυτό. Όταν ολοκληρωθεί ο χρόνος αυτός τότε εμφανίζονται όλοι οι παίκτες που συνδεθήκαν στο παιχνίδι μας στην αριστερή πλευρά της οθόνης μας, όπως παρατηρούμε στην πάνω δεξιά εικόνα του διαγράμματος μας. Στην ίδια εικόνα παρατηρούμε το ζάρι στο κάτω μέρος της οθόνης μας το οποίο είναι ενεργοποιημένο μόνο στον παίκτη του οποίο είναι η σειρά να παίξει. Στην ίδια εικόνα παρατηρούμε στην δεξιά και κάτω πλευρά της οθόνης μας ένα κουμπί με τίτλο Exit το οποίο χρησιμοποιείται στην περίπτωση που κάποιος χρήστης θελήσει να αποχωρήσει από το παιχνίδι. Στην μεσαία και δεξιά εικόνα του διαγράμματος μας βλέπουμε μια κατάσταση του παιχνιδιού στην οποία έχει το παιχνίδι ήδη ξεκινήσει και ο χρήστης έχει ρήξη το ζάρι του και έχει κερδίζει μία κάρτα ενέργειας. Συνεχίζοντας την ροή του παιχνιδιού, στην μεσαία και αριστερή οθόνη μας βλέπουμε την ερώτηση για το αν ο χρήστης μας θέλει να χρησιμοποιήσει μια κάρτα ενέργειας από τις ήδη κερδισμένες και αν θέλει να την εφαρμοστεί στο παιχνίδι. Αυτό μπορεί να συμβεί μόνο στην σειρά του κάθε παίκτη και μόνο αν κάποιες από τις κερδισμένες κάρτες του μπορούν να εφαρμοστούν στο σημείο αυτό. Αν εν συνεχεία, ο παίκτης θελήσει να εφαρμόσει μια κάρτα ενεργεία και επιλέξει το κουμπί OK τότε μεταβαίνει στην φόρμα της οθόνης μας η όποια βρίσκεται στην κάτω αριστερή εικόνα του διαγράμματος μας, και από αυτή μπορεί να επιλέξει μια κάρτα και να την ενεργοποιήσει πατώντας το κουμπί OK. Στην τελευταία οθόνη μας παρατηρούμε την κατάσταση στην οποία το παιχνίδι μας έχει ολοκληρωθεί και εμφανίζεται ένα αναδυόμενο παράθυρο το οποίο μας ενημερώνει για το ποιος είναι ο νικητής του παιχνιδιού και ποια η βαθμολογία αυτού.

Υλοποίηση

Οι τεχνολογίες που θα χρησιμοποιηθούν για τις ανάγκες της δημιουργίας μια εφαρμογής, είτε αυτή είναι διαδικτυακή είτε όχι, θα πρέπει να έχουν μελετηθεί καλά και επίσης καθοριστεί πριν την έναρξη της υλοποίησης, με γνώμονά τις απαιτήσεις αυτής αλλά και τις ήδη υπάρχουσες γνώσεις του ατόμου ή της ομάδας ατόμων οι οποίοι θα επιμεληθούν την υλοποίηση. Σκοπός είναι η παραγωγή του αποδοτικότερου αποτελέσματος στον ταχύτερο δυνατό χρόνο χωρίς υποχωρήσεις από τις αρχικές και ζητούμενες προδιαγραφές.

Ανάλυση απαιτήσεων

Οι τεχνικές και αρχιτεκτονικές απαιτήσεις της εφαρμογής του παιχνιδιού, που μας ζητήθηκε να υλοποιήσουμε, τεθήκαν από εμάς μετά από ανάλυση της δομής αυτού και τον σχεδιασμό του σεναρίου του. Οι απαιτήσεις που προέκυψαν ήταν οι εξής παρακάτω:

- Για αρχή θα πρέπει να υπάρξει στον μηχάνημα που θα βρίσκεται εγκατεστημένο το παιχνίδι, δηλαδή τον εξυπηρετητή (server), ένα πρόγραμμα το οποίο να μπορεί να διαχειριστεί μια διαδικτυακή εφαρμογή και επίσης να έχει εγκατεστημένο ότι είναι απαραίτητο για μπορέσει να γίνει η χρήση του πρωτοκόλλου HTTP.
- Θα πρέπει ακόμη να υπάρξει κάποια βάση δεδομένων η οποία θα αποθηκεύει τα δεδομένα των χρηστών, των παιχνιδιών που έχουν δημιουργήσει ή συμμετέχει αλλά και τα δεδομένα που αλλάζουν κατά την διάρκεια ενός παιχνιδιού. Επίσης η βάση αυτή θα πρέπει να έχει αποθηκευμένο και το στατικό περιεχόμενο του παιχνιδιού το όποιο θα είναι οι ερωτήσεις και οι κάρτες ενεργειών στις οποίες αναφερθήκαμε σε προηγούμενο κεφάλαιο.
- Θα πρέπει επίσης να υλοποιηθεί στον εξυπηρετητή ένα κομμάτι εφαρμογών το οποίο θα ελέγχει όλες τις εργασίες που θα πρέπει να γίνονται στην πλευρά του server. Η χρήση του θα είναι να δέχεται κάποια ερωτήματά από τον χρήστη (client) και θα πρέπει να αποκρίνεται σε αυτά μετά από την υλοποίηση κάποιων εργασιών και υπολογισμών. Το κομμάτι αυτό θα είναι το ίδιο που θα ελέγχει και την επικοινωνία με την βάση δεδομένων. Για παράδειγμα αν ζητηθεί από το παιχνίδι να εμφανιστεί κάποια ερώτηση, θα είναι υποχρέωση του να επικοινωνήσει με την βάση δεδομένων ώστε να μπορέσει να ανασύρει μία ερώτηση από αυτήν και να την επιστρέψει στο παιχνίδι προς εμφάνιση.
- Τρίτο θα πρέπει να υλοποιηθεί ένα αντίστοιχο κομμάτι εφαρμογών το οποίο θα εκτελείται τοπικά στην πλευρά του client, δηλαδή στον φυλλομετρητή (browser), με σκοπό να ελέγχει την πολύπλοκή λογική ενός παιχνιδιού και την υλοποίηση όλων των σεναρίων που έχουν δημιουργηθεί για να εκτελούνται κατά την διάρκεια αυτού.
- Τέταρτο υπάρχει η ανάγκη χρήσης μιας τεχνολογίας η οποία θα μπορεί να προσφέρει συγχρονισμό μεταξύ των χρηστών ενός παιχνιδιού. Δηλαδή όταν για παράδειγμα προχωράει ένας παίκτης πάνω στο ταμπλό του παιχνιδιού, έχοντας κάνει χρήση του

ζαριού, αυτή η κίνηση ταυτόχρονα ή σχεδόν ταυτόχρονα θα πρέπει να παρουσιάζεται σε όσους παίκτες συμμετέχουν στο ίδιο παιχνίδι. Αυτό και άλλα τέτοια παρόμοια σενάρια θα πρέπει με κάποιο τρόπο να γίνονται συγχρονισμένα μεταξύ των παικτών ενός παιχνιδιού.

- Τέλος θα πρέπει να χρησιμοποιηθεί κάποια γλώσσα – τεχνολογία για την παρουσίαση του παιχνιδιού αλλά και όσων προαναφέραμε στην οθόνη του κάθε παίκτη.

Τεχνολογίες που χρησιμοποιήθηκαν

Στο σημείο αυτό θα αναφέρουμε και θα αναλύσουμε τις τεχνολογίες οι οποίες χρησιμοποιήθηκαν για την υλοποίηση του παιχνιδιού.
Έχοντας υπόψη την ανάλυση των απαιτήσεών και γνωρίζοντας ακριβώς τον τελικό στόχο, πάρθηκαν οι αποφάσεις για τις τεχνολογίες με τις οποίες αυτός θα γινόταν εφικτός. Παρακάτω βλέπουμε επιγραμματικά, δίνοντας μια μικρή ανάλυση στην συνέχεια, τις τεχνολογίες που προέκυψαν από την ανάλυση των τεχνολογικών απαιτήσεων.

- Για τον σκοπό της βάσης δεδομένων και της αποθήκευσης σε αυτήν πληροφοριών χρήσιμων για την διεξαγωγή ενός παιχνιδιού πάρθηκε η απόφαση η τεχνολογία που θα χρησιμοποιηθεί να είναι η MySQL για την δομή της βάσης και η SQL (Structured Query Language) να είναι η γλώσσα με την οποία θα υπάρχει η αλληλεπίδραση και επικοινωνία με αυτήν για την ανάκτηση, καταγραφή, αναπροσαρμογή ή διαγραφή δεδομένων της. Ο λόγος της επιλογής είναι διότι η τεχνολογία αυτή είναι ένα πολύ γρήγορο και δυνατό σύστημα διαχείρισης βάσεων δεδομένων και επίσης είναι και η πιο διαδεδομένη για τον σκοπό τον οποίο υπάρχει η ανάγκη μας. (Wikipedia, 2015) (MySql)
- Όσον αφορά την πλευρά του εξυπηρετητή και τις τεχνολογίες που θα χρησιμοποιηθούν για τον σκοπό αυτό, δηλαδή την διαχείριση των ερωτημάτων που θέτει ο client προς τον server, επιλέξαμε τον Apache ο οποίος είναι ένας εξυπηρετητής του παγκοσμίου ιστού. Με αυτήν την τεχνολογία όποτε ένας χρήστης επισκέπτεται μια ιστοσελίδα, το πρόγραμμα πλοήγησης (browser) επικοινωνεί με τον εξυπηρετητή μέσω του πρωτοκόλλου HTTP, ο οποίος συνθέτει την πληροφορία της ζητούμενης ιστοσελίδας και την αποστέλλει πίσω στο πρόγραμμα πλοήγησης. Ο Apache θα είναι υπεύθυνος για την αποστολή της πληροφορίας στον browser αλλά υπεύθυνη για την δημιουργία αυτής θα είναι η γλώσσα προγραμματισμού PHP η οποία θα δέχεται τα ερωτήματα και θα κάνει όλες τις απαραίτητες ενέργειες για την προετοιμασία της ιστοσελίδας ώστε μετά η πληροφορία αυτή να αποσταλεί στον browser μέσω του Apache. Η PHP επίσης θα έχει και τον έλεγχο της επικοινωνίας με την βάση δεδομένων, δηλαδή όποτε ζητάτε κάποια πληροφορία από την βάση δεδομένων θα είναι δική της ευθύνη να δημιουργήσει μιας σύνδεσης με αυτήν ώστε να υπάρξει πρόσβαση στα δεδομένα και ολοκληρώνοντας την όποια ενέργεια, να την τερματίζει. Επίσης επιλέχθηκε και το framework Laravel 4 πάνω στο οποίο βασιστήκαμε για την

δημιουργία όλου του κώδικα της εφαρμογής μας. Στο framework αυτό θα αναφερθούμε πιο λεπτομερώς στην συνέχεια. (Apache) (Wikipedia, 2015) (PHP)

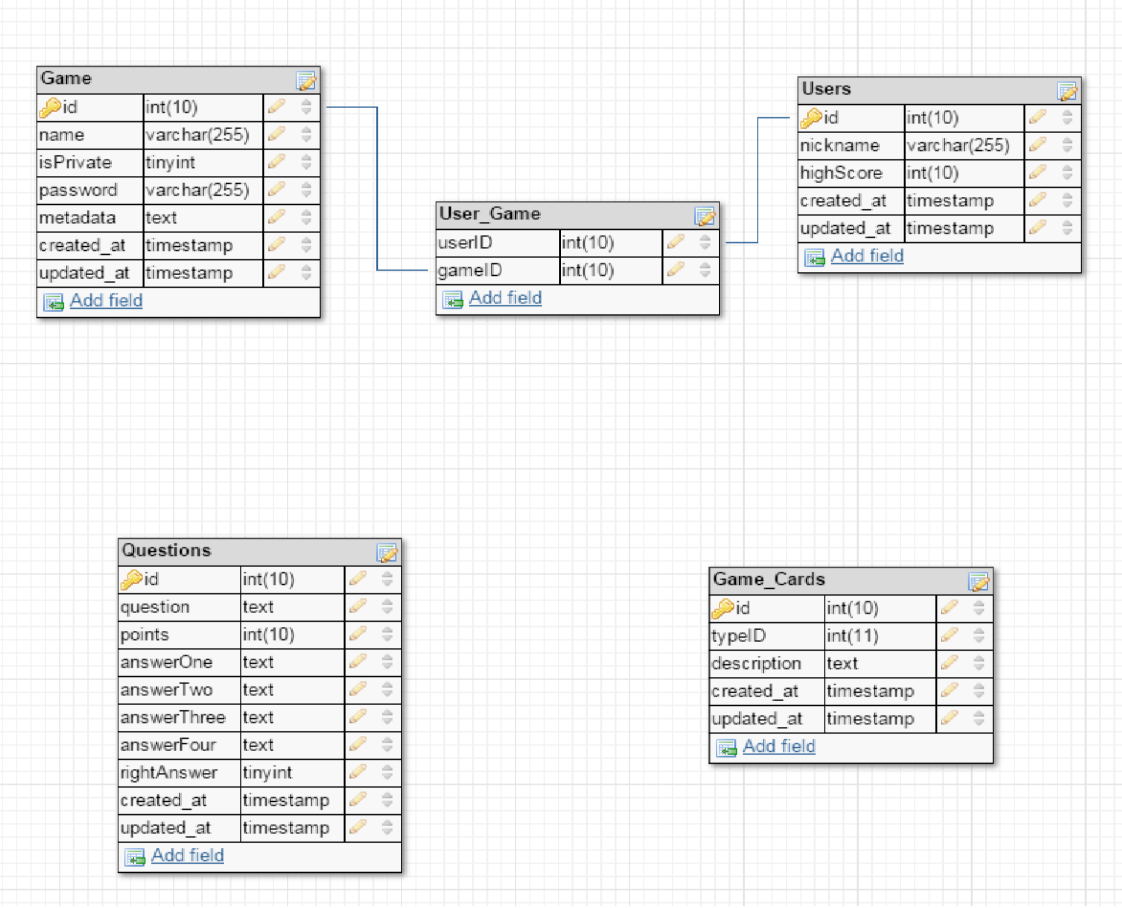
- Αναφερθήκαμε προηγούμενος στην γλώσσα προγραμματισμού η οποία εκτελείται στην πλευρά του εξυπηρετητή. Αναφερθήκαμε, επίσης προηγουμένως, στην ανάλυση των απαιτήσεων για την ανάγκη χρήσης μιας γλώσσα προγραμματισμού η οποία θα εκτελεί εργασίες και σενάρια στην πλευρά του client. Αυτή η γλώσσα είναι η JavaScript και κάποιες βιβλιοθήκες της, τις οποίες θα αναφέρουμε και θα αναλύσουμε παρακάτω. Ο λόγος της επιλογής είναι γιατί είναι η μοναδική γλώσσα που κάνει για την χρήση αυτήν.
- Στην συνέχεια της ανάλυσης των απαιτήσεων προέκυψε το θέμα του συγχρονισμού μεταξύ των browser των παικτών ενός παιχνιδιού. Η επιλογή της τεχνολογίας γι αυτό το σημείο ήταν η πιο δύσκολη αν και είχαμε να διαλέξουμε μόνο μεταξύ δύο, την τεχνολογία των WebSockets και την τεχνολογία Ajax. Η τελική επιλογή είναι η τεχνολογία Ajax. Στην συνέχεια θα εξηγήσουμε τον τρόπο λειτουργίας και των δύο τεχνολογιών αλλά και τον λόγο επιλογής.
- Ολοκληρώνοντας η τεχνολογία για την παρουσίαση του παιχνιδιού στον browser, δηλαδή όλων των δεδομένων που μας έχουν έρθει έτοιμα προς εμφάνιση από τον server, είναι η HTML η οποία είναι και η μοναδική επιλογή που υπάρχει διαθέσιμη. Επίσης για το εικαστικό της παρουσίασης των δεδομένων μας χρησιμοποιήθηκε η τεχνολογία CSS3 βασισμένη στο framework Bootstrap.

Τεχνολογίες Backend

Σε αυτήν την υποενότητα θα παρουσιάσουμε και θα αναλύσουμε με επεξηγήσεις και παραδείγματα τις τεχνολογίες που χρησιμοποιήθηκαν και αναφέρθηκαν παραπάνω, κατά την υλοποίηση του παιχνιδιού όσον αφορά την πλευρά του server. Ξεκινώντας από την τεχνολογία που περικλείει τις υπόλοιπες, εγκαταστήσαμε πριν την έναρξη της υλοποίηση στον υπολογιστή μας το λογισμικό Apache v2.4. Αυτό μας βοηθούσε για όσο διάστημα διήρκησε η υλοποίηση του παιχνιδιού να έχουμε στον υπολογιστή μια προσομοίωση ενός απομακρυσμένου server στον ίδιο μηχάνημα όπου βρίσκεται και ο client. Για παράδειγμα όταν θα ζητούσαμε από τον browser του υπολογιστή μας την τοπική διεύθυνσή "geeky-snake.local", στην οποία έχουμε θέσει προηγούμενος ότι παίζει η εφαρμογή στο μηχάνημα μας για όσο διάστημα διαρκεί η ανάπτυξη της, ο Apache θα ακούσει το ερώτημα (request) και θα κάνει όλες τις απαραίτητες ενέργειες ώστε η απάντηση που θα στείλει πίσω στον browser να είναι αυτό που έχει ζητηθεί σε κάθε περίπτωση. Ακριβώς δηλαδή όπως θα δούλευε ένα απομακρυσμένο request ενός browser σε έναν server.

Δεύτερο βήμα για την υλοποίηση της εφαρμογής μας είναι να δημιουργήσουμε την βάση δεδομένων στην οποία θα αποθηκεύουμε όλη την απαραίτητη πληροφορία για την εύρυθμη λειτουργία της εφαρμογής μας. Εφόσον έχουμε σχεδιάσει στο χαρτί την μορφή ή αλλιώς το σχήμα της βάσης δεδομένων με όλους του πίνακες της, θα πρέπει να εγκαταστήσουμε στο

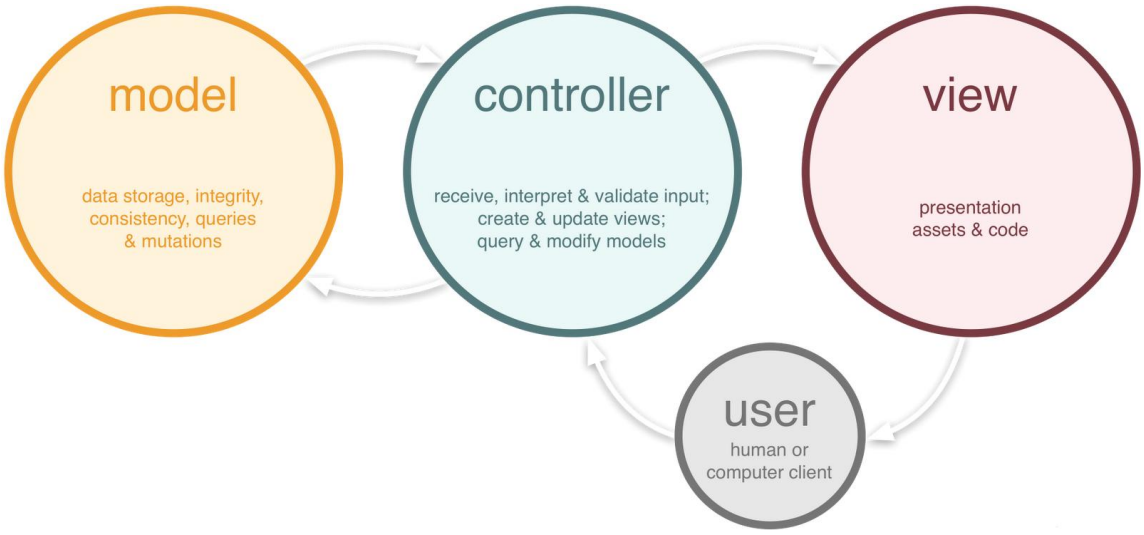
μηχάνημα στο οποίο θα είναι ο server του παιχνιδιού την τεχνολογία SQL για να μπορέσουμε να δημιουργήσουμε την βάση την οποία έχουμε προηγούμενος σχεδιάσει. Η διαδικασία υλοποίησης της βάσης δεδομένων απλοποιείται λόγο του framework Laravel και τον έτοιμων εργαλείων που μας παρέχει τα οποία θα δούμε στην συνέχεια. Επίσης όσον αφορά την βάση δεδομένων εκτός από τους πίνακες που χρησιμοποιούνται για την καταγραφή πληροφοριών των παιχνιδιών και των χρηστών, υπάρχουν και δύο ακόμη πίνακες οι όποιοι περιέχουν το στατικό περιεχόμενο το οποίο αναφέρεται στις ερωτήσεις και στις κάρτες ενεργειών που διαθέτει το παιχνίδι. Για τους δύο αυτούς πίνακες έχουμε φτιάξει κάποια αρχεία τα οποία ονομάζονται seeds και θα πρέπει να εκτελούνται σε κάθε νέα εγκατάσταση της εφαρμογή μας σε κάποιο μηχάνημα εξυπηρετητή έτσι ώστε να γεμίζουν οι πίνακες αυτοί με το απαραίτητο περιεχόμενο. Στο παράρτημα Α υπάρχουν όλα τα βήματα και οι εντολές που χρειάζεται να εκτελεστούν για την σωστή εγκατάσταση της εφαρμογής μας. Στην *εικόνα 27* βλέπουμε σε σχεδιάγραμμα το σχήμα της βάσης δεδομένων που χρησιμοποιεί το παιχνίδι μας.



Εικόνα 27

Προηγουμέ αναφέραμε ότι για την PHP χρησιμοποιήθηκε ένα framework, στην δομή του οποίου βασιστήκαμε και γράψαμε όλο τον κώδικα της εφαρμογής μας. Το framework αυτό είναι βασισμένο στο σχεδιαστικό μοντέλο MVC (Model – View – Controller) το οποίο

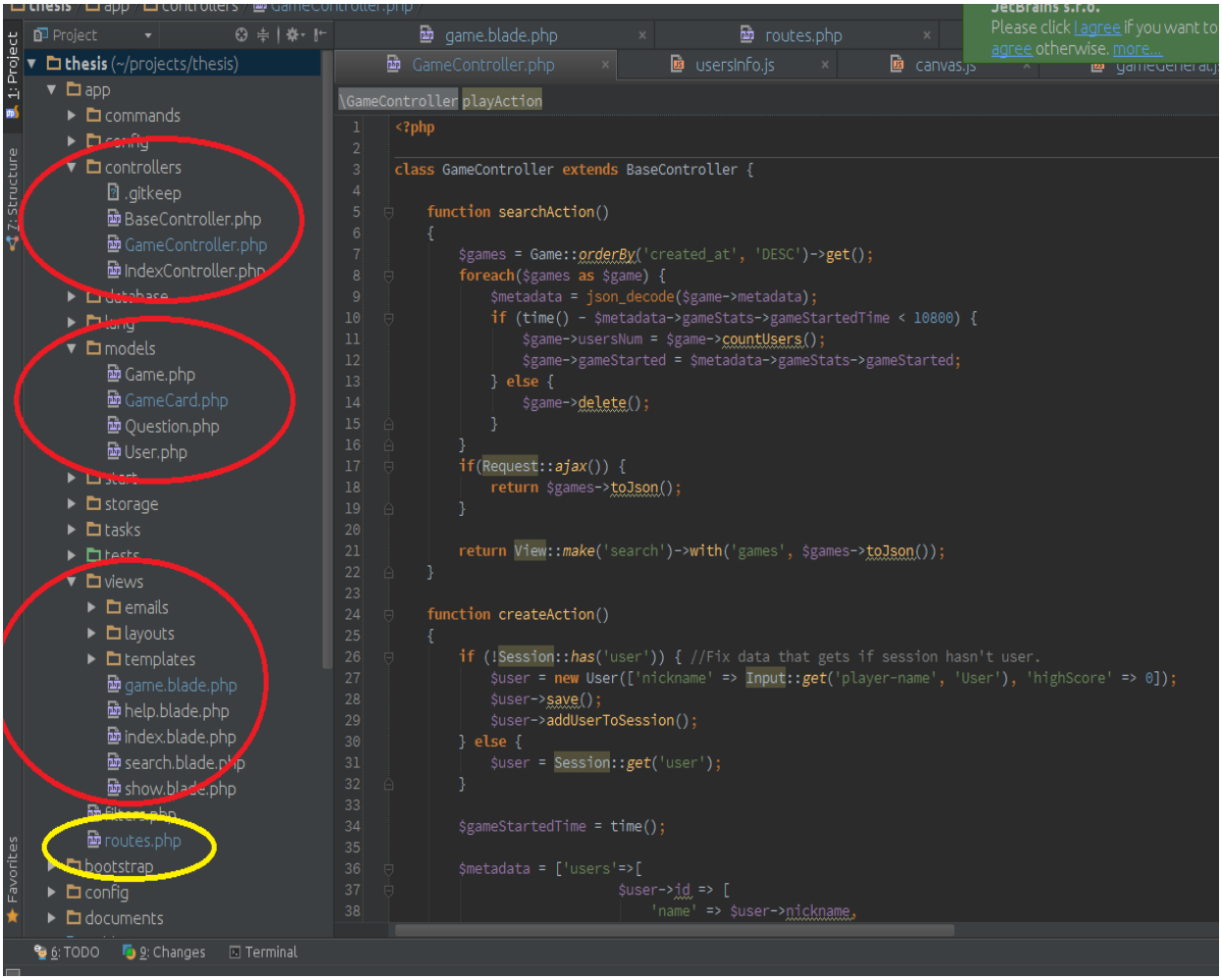
εξηγούμε στην συνέχεια. Γενικά η χρήση των frameworks κάνει την δουλειά των προγραμματιστών πιο εύκολη και δομημένη ειδικότερα στην υλοποίηση μεγάλων έργων. Το framework μας παρέχει μια σειρά από έτοιμες λειτουργίες, αυτές που είναι πιο σύνηθες να χρησιμοποιούνται όπως είναι για παράδειγμα η σύνδεση με την βάση δεδομένων ή η αλλαγή σε έναν πίνακα ή ακόμη η πρόσθεση ενός νέου στην βάση δεδομένων, κάτι το οποίο χρειάζεται να συμβαίνει συχνά κατά την υλοποίηση και την λειτουργία μιας εφαρμογής. Οι παραπάνω λειτουργίες παρέχονται έτοιμες από το framework χωρίς να χρειάζεται οι προγραμματιστές να γράφουν αρκετές γραμμές κώδικα για τις επαναλαμβανόμενες αυτές λειτουργίες. Επίσης το framework παρέχει μια πιο δομημένη τοποθέτηση των αρχείων μέσα σε φακέλους με σκοπό την ευκολότερη εύρεση κατά την αναζήτηση, αλλά και την τμηματοποίηση αυτών, μέσα σε μεγάλες και πολύπλοκες εφαρμογές. Για την καλύτερη δόμηση και ιεράρχηση των αρχείων μέσα στο framework βοηθάει και το σχεδιαστικό μοντέλο MVC, το οποίο προαναφέραμε, σπάζοντας τα αρχεία της εφαρμογής τμηματικά ανάλογος την λειτουργικότητα τους. Τα αρχεία, στα frameworks που χρησιμοποιούν το σχεδιαστικό μοντέλο MVC, είναι κατά κανόνα χωρισμένα πρώτον σε Controllers, τα οποία είναι αρχεία οπού ελέγχουν και δέχονται όλα τα request (endpoints) που μπορεί να γίνουν από τον client, και επιστρέφουν σε αυτόν το ζητούμενο κάθε φορά αποτέλεσμα. Δεύτερον σε Models τα οποία είναι τα αρχεία που έχουν την δυνατότητα να κάνουν όλους του απαιτούμενους υπολογισμούς, να εκτελούν του αλγόριθμούς, να ελέγχουν την επικοινωνία με την βάση δεδομένων και να δημιουργούν την πληροφορία σε τέτοια μορφή η οποία να είναι έτοιμη να ληφθεί από την τελευταία και τρίτη κατηγορία αρχείων τα οποία ονομάζονται Views και περιέχουν την γλώσσα παρουσίασης στον browser HTML και κάποιες φορές PHP ή και JavaScript. Εφόσον είναι έτοιμες και τοποθετημένες όλες οι πληροφορίες μέσα στο HTML αρχείο και αυτό είναι έτοιμο για αποστολή στον browser προς εμφάνιση, αναλαμβάνει ο Controller να αποστείλει το μορφοποιημένο αυτό HTML αρχείο, δηλαδή να απαντήσει στο request που δέχθηκε. Στην *εικόνα 28* παρατηρούμε την περιγραφή που έγινε για το MVC σχεδιαστικό μοντέλο σε γράφημα ακολουθίας. (Wikipedia, 2015)



Εικόνα 28

Αυτός είναι ο γενικός και στην πλειοψηφία των περιπτώσεων απαράβατος κανόνας του τρόπου λειτουργίας του framework Laravel με την χρήση του σχεδιαστικού μοντέλου MVC. Αν δεν ακολουθηθούν οι παραπάνω κανόνες για την δημιουργία των αρχείων που εμπεριέχουν τον κώδικα της εφαρμογής μας και η εφαρμογή αρχίσει να μεγαλώνει με γρήγορους ρυθμούς τότε ο κώδικάς αρχίζει και περιπλέκεται και γίνεται πολύ δύσκολος στην ανάγνωση και την κατανόηση.

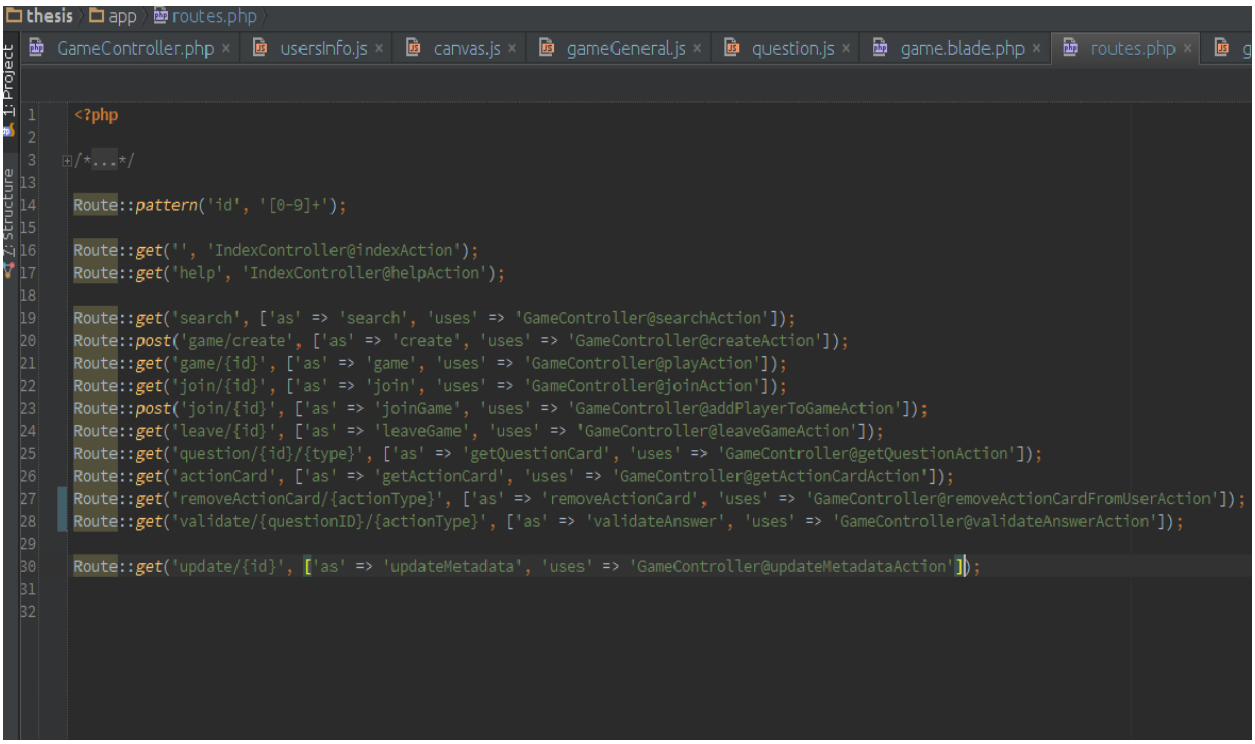
Στην *εικόνα 29* παρατηρούμε ένα τμήμα της δομής των αρχείων της εφαρμογή μας μέσω του IDE (Integrated Development Environment) που χρησιμοποιήθηκε για την υλοποίηση, σε αυτήν έχουμε κυκλώσει με κόκκινο χρώμα τα τρία σημαντικά σημεία του σχεδιαστικού προτύπου MVC που αναφέραμε προηγούμενος στα οποία φαίνονται τα αρχεία και οι υποφακέλοι τους οποίους περιέχουν (Wikipedia, 2015).



Εικόνα 29

Επίσης στην *εικόνα 29* παρατηρούμε στην αριστερή κάτω πλευρά ένα αρχείο το οποίο είναι σηματοδοτημένο με κίτρινο χρώμα και ονομάζεται routes.php. Του αρχείου αυτού τα περιεχόμενα βλέπουμε στην *εικόνα 30*. Στο αρχείο αυτό υπάρχουν όλα τα URI's (Unique Resource Identifiers) στα οποία μπορεί να "ακούσει" η εφαρμογή και να εκτελέσει κάποιες ενέργειες. Αν δεν δηλωθεί σε αυτό το αρχείο κάποιο URI το οποίο θέλουμε την ύπαρξη του, τότε καλώντας το από τον browser θα μας εμφανίζει την ένδειξη ότι αυτή η διεύθυνση που ζητήθηκε δεν υπάρχει. (Wikipedia, 2015)

Σε κάθε URI που δηλώνουμε, όπως παρατηρούμε και στην εικόνα, αναφέρουμε την μέθοδο πρωτοκόλλου HTTP που θα ανταποκρίνεται το καθένα, δηλαδή στην περίπτωση μας αν θα είναι POST ή GET, επίσης το URI θα δείχνει ποιο το σημείο του κώδικα το οποίο θα πρέπει να εκτελεστεί σε κάθε κλήση σε αυτό. Για παράδειγμα στην γραμμή 20 υπάρχει το URI "game/create" το οποίο αντιστοιχεί στο URL www.geeky-snake.com/game/create αυτό ανταποκρίνεται μόνο όταν καλεστεί με την μέθοδο POST που βλέπουμε ότι είναι δηλωμένη στο αρχή της γραμμής, αυτό σημαίνει ότι η εργασία που θα εκτελέσει ο κώδικάς που εφαρμόζεται σε αυτό το request είναι να εγγράψει στην βάση δεδομένων τις πληροφορίες ενός νέου παιχνιδιού. Και ο κώδικας που θα εκτελεστεί και θα ξεκινήσει την εργασία εγγραφής ξεκινάει από το αρχείο GameController και την συνάρτηση createAction() που επίσης βλέπουμε στο τέλος της δήλωσης της γραμμής 20. (Wikipedia, 2015)



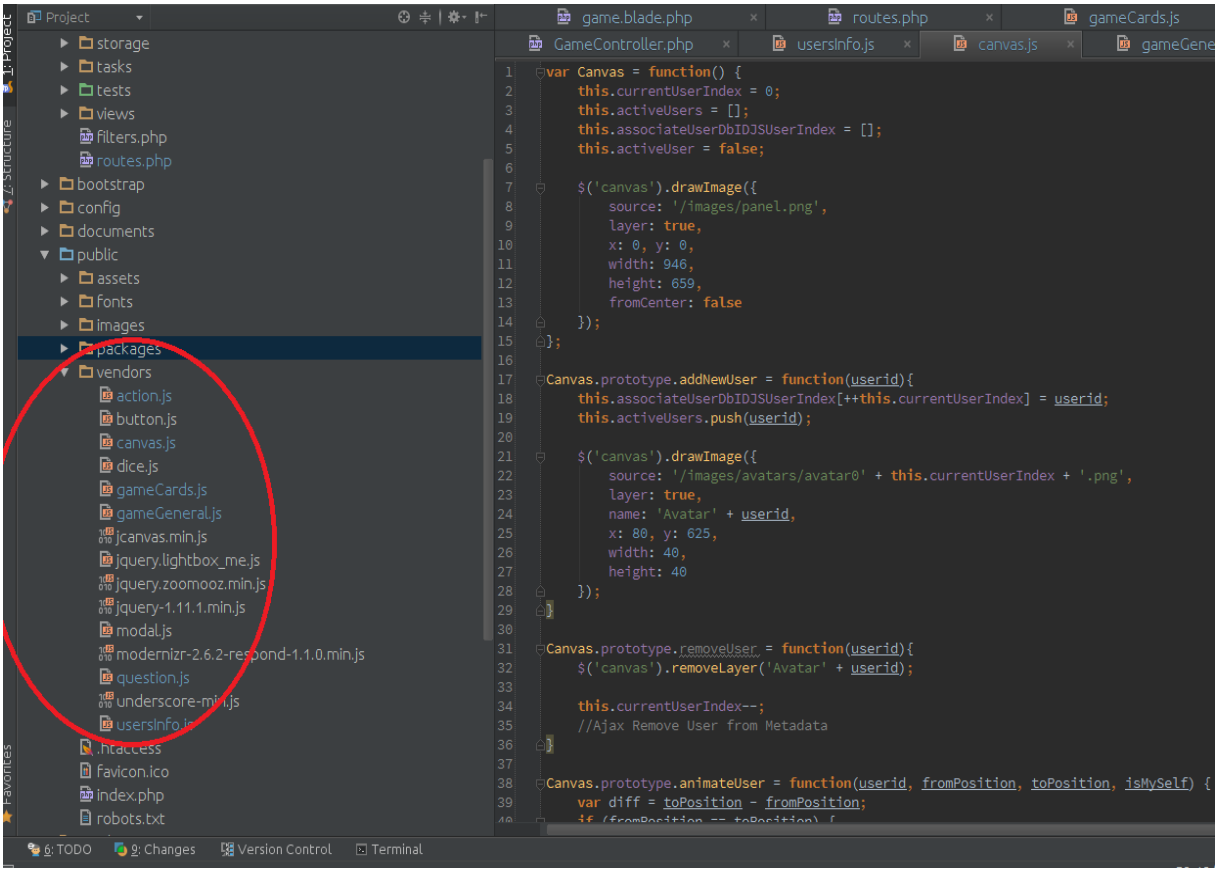
Εικόνα 30

Στο παράρτημα 2 της εργασίας μας βρίσκονται διαθέσιμα όλα τα αρχεία με όλο τον κώδικα που έχει γραφτεί για τις ανάγκες του παιχνιδιού.

Τεχνολογίες Frontend

Στο υποκεφάλαιο αυτό θα αναφερθούμε στις τεχνολογίες που χρησιμοποιούνται στην πλευρά του χρήστη (client) και διαχειρίζονται την σωστή και ομαλή παρουσίαση του παιχνιδιού. Για τον λόγο αυτό έχει δοθεί και η ονομασία Frontend.

Ξεκινώντας με την JavaScript η οποία είναι μια γλώσσα προγραμματισμού οπού βρίσκει κυρίως εφαρμογή στους browsers και χρησιμοποιείται από αυτούς για να τους δίνεται η δυνατότητα να εκτελούν συγκεκριμένα σενάρια, να μπορούν να επικοινωνούν με τον server της εφαρμογής ασύγχρονα και να ανταλλάσουν δεδομένα με αυτούς χωρίς να επηρεάζεται η σελίδα που βλέπει ο χρήστης αν εμείς δεν το θελήσουμε αυτό. Επίσης μέσω της JavaScript μας δίνεται η δυνατότητα να ελέγχουμε δυναμικά το περιεχόμενο που εμφανίζεται στον χρήστη. Εμείς χρησιμοποιήσαμε και εργαστήκαμε με την πιο βασική βιβλιοθήκη της JavaScript, όπου ονομάζεται JQuery, και είναι σχεδιασμένη με τέτοιο τρόπο ώστε να απλοποιεί κατά πολύ την υλοποίηση σεναρίων μέσω του κώδικά. (jQuery)
Λόγο των μεγάλων αναγκών για τα σενάρια χρήσης, μιας εφαρμογής ενός παιχνιδιού, δημιουργήθηκε σε γλώσσα JavaScript αρκετός κώδικας ο οποίος διαχωρίστηκε σε μικρά αρχεία με σημασιολογική ονομασία για το καθένα έτσι ώστε να μπορεί ο οποιοσδήποτε να καταλαβαίνει ποια είναι η αποστολή του κώδικά που βρίσκεται στο εκάστοτε αρχείο.
Στην *εικόνα 31* βλέπουμε τα αρχεία JavaScript στα οποία αναφερθήκαμε προηγούμενος, επίσης παρατηρούμε ότι τα αρχεία αυτά περιέχονται εξωτερικά από έναν φάκελο ο οποίος ονομάζεται public. Ότι αρχείο περιέχεται μέσα σε αυτόν τον φάκελο είναι αρχείο το οποίο αποστέλλεται στον browser από τον server και εκτελείται ή εμφανίζεται σε αυτόν. Μέσα σε αυτόν τον φάκελο υπάρχουν ακόμη, τα .css αρχεία τα οποία συντελούν στην εικαστική μορφοποίηση της σελίδας μας και για τα οποία θα μιλήσουμε στην συνέχεια, επίσης υπάρχουν και όλες οι εικόνες που χρησιμοποιούνται στο παιχνίδι, οι γραμματοσειρές τις οποίες χρησιμοποιούν τα κείμενα μας για να εμφανιστούν και άλλα διάφορα αρχεία τα οποία δεν θα αναφερθούμε γιατί δεν έχουν κάποιο τεχνολογικό ενδιαφέρων όσον αφορά το παιχνίδι και την υλοποίηση του.

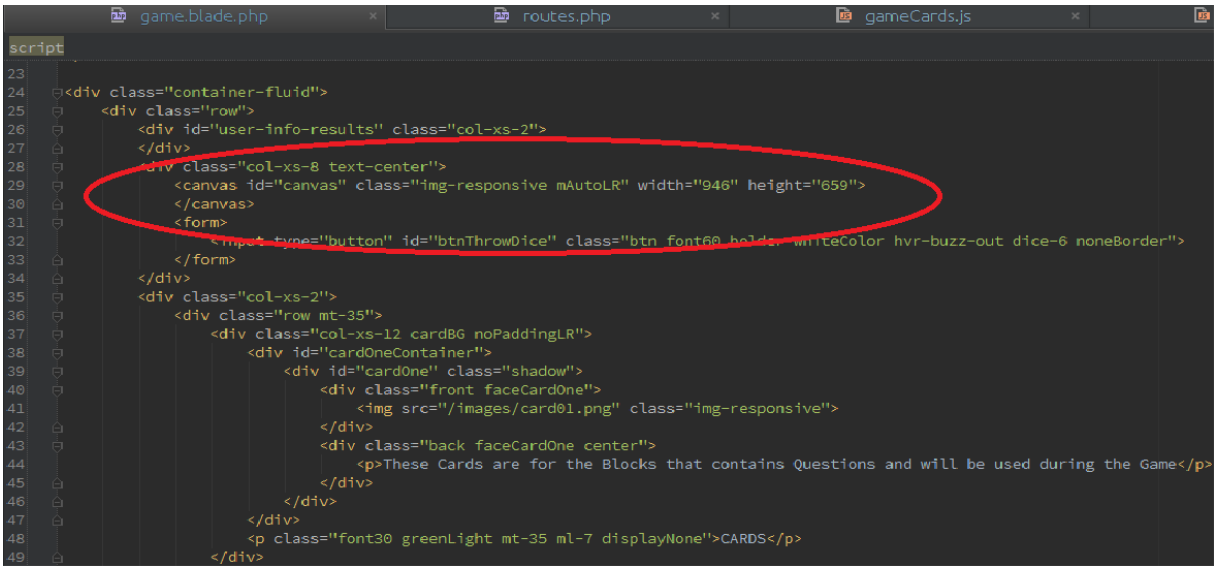


Εικόνα 31

Μέσα στον φάκελο vendors που βλέπουμε στην *εικόνα 28* υπάρχουν τα αρχεία JavaScript τα οποία δημιουργήσαμε για τις ανάγκες του παιχνιδιού μας, επίσης υπάρχουν και οι βιβλιοθήκες τις οποίες χρησιμοποιούμε για την υλοποίηση του παιχνιδιού μας. Αυτές είναι οι εξής jcanvas, jquery-lightbox, jquery-zoomooz, jquery και τέλος underscore στην συνέχεια θα εξηγήσουμε τον λόγο χρήσης της καθεμίας από αυτές.

Στην βιβλιοθήκη jquery αναφερθήκαμε προηγούμενος στον σκοπό και την χρήση της, θα συνεχίσουμε με τις υπόλοιπες βιβλιοθήκες ξεκινώντας από την jcanvas η οποία είναι μια βιβλιοθήκη που πρέπει να συνυπάρχει με την έκδοση 5 της γλώσσα σήμανσης και μορφοποίησης υπερκειμένου HTML στην οποία θα αναφερθούμε πιο αναλυτικά, η γλώσσα αυτή στην έκδοση 5 διαθέτει μια νέα ετικέτα σήμανσης η οποία ονομάζεται canvas και χρησιμοποιείται για τον σχεδιασμό γραφικών σε πραγματικό χρόνο μέσω της χρήσης JavaScript σε μια ιστοσελίδα. (Canvas) Από την στιγμή που εμφανίστηκε αυτή η ετικέτα στην καινούργια έκδοση της HTML έχει αντικαταστήσει σε πάρα πολλές ιστοσελίδες την τεχνολογία του flash οπού επίσης χρησιμοποιείται για την δημιουργία γραφικών. Εμείς χρησιμοποιούμε την ετικέτα αυτή για να παρουσιάσουμε το ταμπλό του παιχνιδιού μας και οποιαδήποτε ενεργεία ή κίνηση γίνεται πάνω σε αυτό. Στην *εικόνα 32* βλέπουμε το σημείο στον HTML κώδικα μας το οποίο υπάρχει η ετικέτα canvas η οποία δεν έχει κάποιο περιεχόμενο. Έχει δηλωμένο παρόλα αυτά κάποιο σταθερό μέγεθος μέσω τον παραμέτρων width και height. Αυτός είναι ο μέγιστος χώρος ο οποίος θα καταλάβει αν εμφανιστεί σε μία οθόνη μεγάλων διαστάσεων. Παρόλη την δηλωμένη μέγιστη διάσταση, αν το μέγεθος του

παραθύρου μας συρρικνωθεί τότε η εικόνα που εμφανίζεται μέσα στην ετικέτα μικραίνει και αυτή λόγο της χρήσης μιας τεχνολογίας που ονομάζεται responsive την οποία θα δούμε στην συνέχεια.



Εικόνα 32

Στην *εικόνα 33* βλέπουμε το αρχείο το οποίο περιέχει όλη την υλοποίηση των σεναρίων που γίνονται πάνω στην canvas ετικέτα με την χρήση των βιβλιοθηκών jquery και jcanvas. Η ετικέτα canvas είναι αυτή στην οποία προσαρμόζεται η εικόνα του ταμπλό του παιχνιδιού κατά την έναρξή του και πάνω σε αυτήν δυναμικά προστίθενται, αφαιρούνται και κινούνται οι παίκτες. (Jcanvas)

Η πρώτη συνάρτηση η οποία περιέχεται στην *εικόνα 33* και λέγεται drawImage είναι αυτή που κατά την κλήση της είναι υπεύθυνη για την εμφάνιση της εικόνας του ταμπλό μέσα στο παράθυρο του παιχνιδιού σε σταθερές διαστάσεις. Οι επόμενες δύο με ονόματα addNewUser και removeUser, η πρώτη είναι υπεύθυνη για την προσθήκη ενός νέου χρήστη και του avatar αυτού πάνω στο ταμπλό του παιχνιδιού και η δεύτερη είναι υπεύθυνη για την απομάκρυνση του avatar του από το ταμπλό σε περίπτωση εξόδου, του παίκτη στον οποίο ανήκει, από το παιχνίδι χρησιμοποιώντας το κουμπί "Exit". Όλες οι εικόνες που προστίθενται πάνω στην ετικέτα canvas βρίσκονται σε διαφορετικό επίπεδο η μία από την άλλη έχοντας και διαφορετικό όνομα. Με αυτόν τον τρόπο η μία είναι ανεξάρτητη από την άλλη και η κίνηση της μίας δεν επηρεάζει τις υπόλοιπες.
Επίσης τέλος υπάρχει η συνάρτηση με όνομα animateUser η οποία είναι υπεύθυνη για το σενάριο της μετακίνησης ενός παίκτη πάνω στο ταμπλό του παιχνιδιού κάθε φορά που αυτός χρησιμοποιήσει το ζάρι ή κάνει χρήση κάποιας κάρτας ενεργείας η οποία του επιτρέπει να κινηθεί πάνω στο ταμπλό χωρίς την ρήψη του ζαριού.

canvas.js

```
var Canvas = function() {
    this.currentUserIndex = 0;
    this.activeUsers = [];
    this.associateUserDbIDJSUserIndex = [];
    this.activeUser = false;

    $('canvas').drawImage({
        source: '/images/panel.png',
        layer: true,
        x: 0, y: 0,
        width: 946,
        height: 659,
        fromCenter: false
    });
};

Canvas.prototype.addNewUser = function(userid){
    this.associateUserDbIDJSUserIndex[++this.currentUserIndex] = userid;
    this.activeUsers.push(userid);

    $('canvas').drawImage({
        source: '/images/avatars/avatar0' + this.currentUserIndex + '.png',
        layer: true,
        name: 'Avatar' + userid,
        x: 80, y: 625,
        width: 40,
        height: 40
    });
}

Canvas.prototype.removeUser = function(userid){
    $('canvas').removeLayer('Avatar' + userid);

    this.currentUserIndex--;
    //Ajax Remove User from Metadata
}

Canvas.prototype.animateUser = function(userid, fromPosition, toPosition, isMySelf) {
    var diff = toPosition - fromPosition;
    if (fromPosition == toPosition) {
        return;
    }

    if (fromPosition <= toPosition) {
        for (var i = fromPosition; i <= toPosition; i++) {
            $('canvas').animateLayer('Avatar' + userid, boardData[i].position);

            if(i == toPosition && isMySelf && window.gameMetadata.users[userid].actionCards.length == 0) {
                console.log("NoAction");
                $(document).trigger('blockEvent', [toPosition, userid, diff]);
            } else if (i == toPosition && isMySelf && window.gameMetadata.users[userid].actionCards.length > 0) {
                console.log("Action");
                $(document).trigger('askForAction', [toPosition, userid, diff]);
            }
        }
    } else if (fromPosition >= toPosition && !isMySelf) {
```

Εικόνα 33

Προηγουμένως στην εργασία μας αναφερθήκαμε στο τεχνολογικό κομμάτι που αφορά τον συγχρονισμό των κινήσεων και των ενεργειών μεταξύ των παικτών ενός παιχνιδιού. Γι αυτή μας την απόφαση, δηλαδή το ποία τεχνολογία μεταξύ των Ajax και WebSockets θα χρησιμοποιούντο έπαιξε ρόλο η γλώσσα προγραμματισμού που είχαμε ήδη επιλέξει να χρησιμοποιήσουμε και η οποία δεν είναι αρκετά απλή στην συνεργασία της με την τεχνολογία των WebSockets, ο δεύτερος και επίσης σημαντικός λόγος ήταν ότι ήδη γνωρίζαμε αρκετά την τεχνολογία και τον τρόπο χρήσης του Ajax σε σύγκρισή με τις γνώσεις μας για τις ανάγκες μια υλοποίησης του παιχνιδιού με την χρήση της τεχνολογίας των WebSockets. Αυτοί οι δύο λόγοι μας προέτρεψαν να επιλέξουμε για την υλοποίηση την τεχνολογία Ajax. (Wikipedia, 2015) (WebSockets) (Wikipedia) (w3schools, 2015)

Θα εξηγήσουμε τον τρόπο χρήσης με απλά λόγια και για τις δύο τεχνολογίες για να γίνει κατανοητή και ποια η διαφορά μεταξύ τους. Στην πρώτη περίπτωση των WebSockets η ανταλλαγή πληροφορίας του client με τον server γίνεται μέσω ενός διαύλου επικοινωνίας ο οποίος παραμένει ανοικτός για όσο χρόνο καθορίσουμε. Για να ξεκινήσει και να εγκατασταθεί αυτό το κανάλι επικοινωνίας γίνεται ένα request από τον browser στον server και μόλις αποδεκτεί το αίτημα ο τελευταίος εγκαθιδρύεται το κανάλι επικοινωνίας. Οτιδήποτε αλλάζει στα δεδομένα του socket που βρίσκεται στον server αυτόματος ενημερώνονται και όλοι οι clients που είναι συνδεδεμένοι σε αυτό. Στην περίπτωση του παιχνιδιού μας αυτός θα ήταν και ο τρόπος ανταλλαγής των ενημερώσεων και των αλλαγών που συμβαίνουν στο παιχνίδι κατά την διάρκεια του από τις ενέργειες των παικτών. Δηλαδή όλοι οι παίκτες ενός στιγμιότυπου του παιχνιδιού μας θα ήταν συνδεδεμένοι σε ένα socket στον server και θα αντάλλαζαν πληροφορίες μέσω αυτού.

Ο τρόπος που τελικά χρησιμοποιήθηκε στο παιχνίδι μας, είναι αυτός με την χρήση της τεχνολογίας Ajax, ο οποίος είναι αρκετά παρόμοιος ως προς το τελικό αποτέλεσμα με τα WebSockets αλλά έχει αρκετές τεχνικές διαφορές. Οι πληροφορίες του παιχνιδιού, οι οποίες ονομάζονται metadata, αποθηκεύονται στην βάση δεδομένων στον πίνακα Game και πεδίο metadata όπως φαίνεται και στο σχήμα της βάσης στην *εικόνα 27.* Δηλαδή όλες οι αλλαγές που γίνονται από οποιοδήποτε παίκτη στην διάρκεια ενός στιγμιότυπου του παιχνιδιού αποθηκεύονται σε αυτό το σημείο. Ο τρόπος λειτουργίας είναι ο εξής, κάθε αλλαγή που συμβαίνει στον browser ενός παίκτη και συσχετίζεται με τις πληροφορίες του παιχνιδιού, αμέσως δημιουργείτε ένα request προς τον server έτσι ώστε να ενημερωθούν τα metadata του παιχνιδιού. Παράλληλα κάθε αρκετά μικρό χρονικό διάστημα στέλνεται ένα request από τον browser όλων των παικτών ενός παιχνιδιού στον server έτσι ώστε να ελέγξουν για τυχόν αλλαγές σε αυτό. Αν κατά την αποστολή ενός request βρεθούν αλλαγές στα metadata του παιχνιδιού, σε σύγκριση με τα τελευταία, τότε αυτές εφαρμόζονται σε όλους τους χρήστες που εμπλέκονται στο παιχνίδι αυτό.

Η βασική διαφορά του Ajax με τα WebSockets είναι ότι το πρώτο κάθε φορά που χρειάζεται να επικοινωνήσει με τον server πρέπει να δημιουργήσει μια νέα σύνδεση η οποία παραμένει ανοικτή μέχρι το τέλος της επικοινωνίας, αυτό αυξάνει τον όγκο της πληροφορίας που ανταλλάσσεται σε κάθε request, σε σύγκριση με τον τρόπο λειτουργίας της τεχνολογίας των WebSockets, για να μπορεί να καθίσταται δυνατή η δημιουργία μιας νέας σύνδεσης. Αυτό έχει σαν αποτέλεσμα αν αυξηθεί υπερβολικά η αποστολή request στον server στον οποίο είναι εγκατεστημένη η εφαρμογή μας να δημιουργήσει προβλήματα αντοχής στην

επισκεψιμοτήτα και στον όγκο των παιχνιδιών τα οποία μπορεί να διαχειρίζεται ταυτόχρονα, κάτι το οποίο δεν μας επηρεάζει, ούτε πρόκριτε να μας επηρεάσει στο κοντινό μέλλον. Η τεχνολογία του Ajax εκτός από τα θετικά που μας προσέφερε μας δημιούργησε και κάποια προβλήματα τα οποία έπρεπε να διαχειριστούμε. Το πρώτο ήταν ότι η παράλληλη αποστολή πολλών ασύγχρονων request προς τον server μας και εν συνεχεία προς την βάση δεδομένων θα μπορούσε να προκαλέσει την μη σωστή ενημέρωση των δεδομένων σε όλους του χρήστες. Αυτό συμβαίνει διότι εκτός από την αποστολή request από τους χρήστες ανά τακτά χρονικά διαστήματα για την ενημέρωση της κατάστασης του παιχνιδιού, στέλνονται και κάποια ασύγχρονα request σε διάφορες ενέργειες χρηστών, όπως για παράδειγμα η απάντηση σε μία ερώτηση ή η χρήση μιας κάρτας ενεργείας. Αυτό μπορεί να προκαλέσει κάποια σύγχυση στον ταυτόχρονο συγχρονισμό των δεδομένων, και για ένα αρκετά μικρό διάστημα θα μπορούσε όλοι οι χρήστες να μην βλέπουν τα ίδια δεδομένα για το παιχνίδι στους browsers τους. Αυτό λύθηκε με την χρήση του πρότυπου κλειδώματος του πίνακα της βάσης, του οποίου τον κώδικα βλέπουμε στην *εικόνα 34*.

```
function getActionCardAction() {
    $user = Session::get('user');
    do {
        $game = Game::find(Session::get('gameID'));
        sleep(0.2);
    } while($game->lock);

    $game->lock = true;
    $game->save();
    $gameMetadata = json_decode($game->metadata);
    if (isset($gameMetadata->users->{$user->id})) {
        $metadataUser = $gameMetadata->users->{$user->id};
            if (count($metadataUser->actionCards) == 9) {
                $actionCard = GameCard::firstOrFail();
                $metadataUser->actionCards[] = $actionCard->typeID;
            } else {
                $actionCard = GameCard::whereNotIn('typeID', $metadataUser->actionCards)->orderByRaw("RAND()")->first();
                $metadataUser->actionCards[] = $actionCard->typeID;
            }
    }
    $game->metadata = json_encode($gameMetadata);
    $game->lock = false;
    $game->save();
    return $actionCard;
}
```

Εικόνα 34 Lock Database

Η λογική έχει ως εξής, δεν θα μπορούν να γράφουν στην βάση δεδομένων πολλά requests ταυτόχρονος αλλά θα δημιουργείτε μια σειρά προτεραιότητας. Αυτό θα μπορούσε να σκεφτεί κάποιος ότι θα δημιουργούσε μια τεράστια σειρά προτεραιότητας λόγο των πολλαπλών ασύγχρονων request, όμως αυτό δεν συμβαίνει γιατί στην πραγματικότητα μόνο τα request που έχουν κάποια διαφορετική πληροφορία, από την ήδη υπάρχουσα της βάσης δεδομένων, γράφουν σε αυτή και έτσι και πετυχαίνουμε την σειριακή τοποθέτηση της πληροφορίας στην βάση δεδομένων και δεν αυξάνουμε τους χρόνους απόκρισης των request.
Επίσης μια ακόμη μέθοδος η οποία χρησιμοποιήθηκε για την γρηγορότερη απόκριση των request είναι η μη εκτέλεση της μορφοποίησης του κώδικα HTML, που εξηγούμε ακριβώς

από κάτω την λειτουργία του, σε περίπτωση που δεν έχει αλλάξει κάποια από τις πληροφορίες μας στην απάντηση του request σε σύγκριση με αυτές που έχει ήδη ο χρήστης στην διάθεση του. Ο τρόπος με τον οποίο το πετυχαίνουμε αυτό είναι, η τρέχουσα πληροφορία η οποία επιστρέφει στον χρήστη μας έχει κωδικοποιηθεί με έναν αλγόριθμο που λέγεται md5 κατά την αποστολή της από τον browser στον client (βλέπε εικόνα 35)

```
        if (isset($gameMetadata->users->{$user->id})) {
            $metadataUser = $gameMetadata->users->{$user->id};

            $isNewPositionSmaller = $metadataUser->currentPosition > $jumpTo;
            $metadataUser->currentPosition = $jumpTo;
            $metadataUser->coordinates = Game::$squares[$metadataUser->currentPositi
            if (is_null($pointFromActionCard)) {
                $metadataUser->score = $isNewPositionSmaller ? $metadataUser->score
            } else {
                $metadataUser->score += $pointFromActionCard;
            }
            $gameMetadata->users->{$user->id} = $metadataUser;
        }
    }

    $game->metadata = json_encode($gameMetadata);
    $game->lock = false;
    $game->save();

    return ['metadata' => $gameMetadata, 'status' => md5($game->metadata)];
}
```

Εικόνα 35 Create Game Status

Ο αλγόριθμος αυτός παράγει μια συστοιχία 128bit, αν η συστοιχία αυτή που παράγεται σε ένα request που γίνεται για τον έλεγχο τυχόν αλλαγών στις πληροφορίες του παιχνιδιού μας είναι ίδια με την ήδη υπάρχουσα δεν εκτελείται καμιά ενέργεια στον κώδικα της HTML. Αυτός ο έλεγχος γίνεται στην πλευρά του client μέσω της JavaScript όπως παρατηρούμε στην *εικόνα 36*.

```
var UsersInfo = function(canvasInstance){
    var lastUsersPosition = [],
        lastCurrentStatus = '';

    loadUsersInfo = function(data) {

        if (lastCurrentStatus == data.status) {
            return;
        }
        lastCurrentStatus = data.status;

        window.gameMetadata = data.metadata;

        _(data.metadata.users).each(function (userData, userID) {
            if (lastUsersPosition[userID] >= 50) {
                return;
            }

            var isCurrentPositionAtTheEnd       = (userData.currentPosition >= 50),
                isCurrentPositionAtTheBegging   = (userData.currentPosition == 0),
                isMySelf                        = (userID == userIDFromSession),
                isUsersTurn                     = (userData.activeUser == true),
                isUserNotExistsInGame           = (canvasInstance.activeUsers.indexOf(userID) === -1),
                isInSameLocation                = (lastUsersPosition[userID] == userData.currentPosition),
                isBeforeDiceRole                = (isMySelf);
```

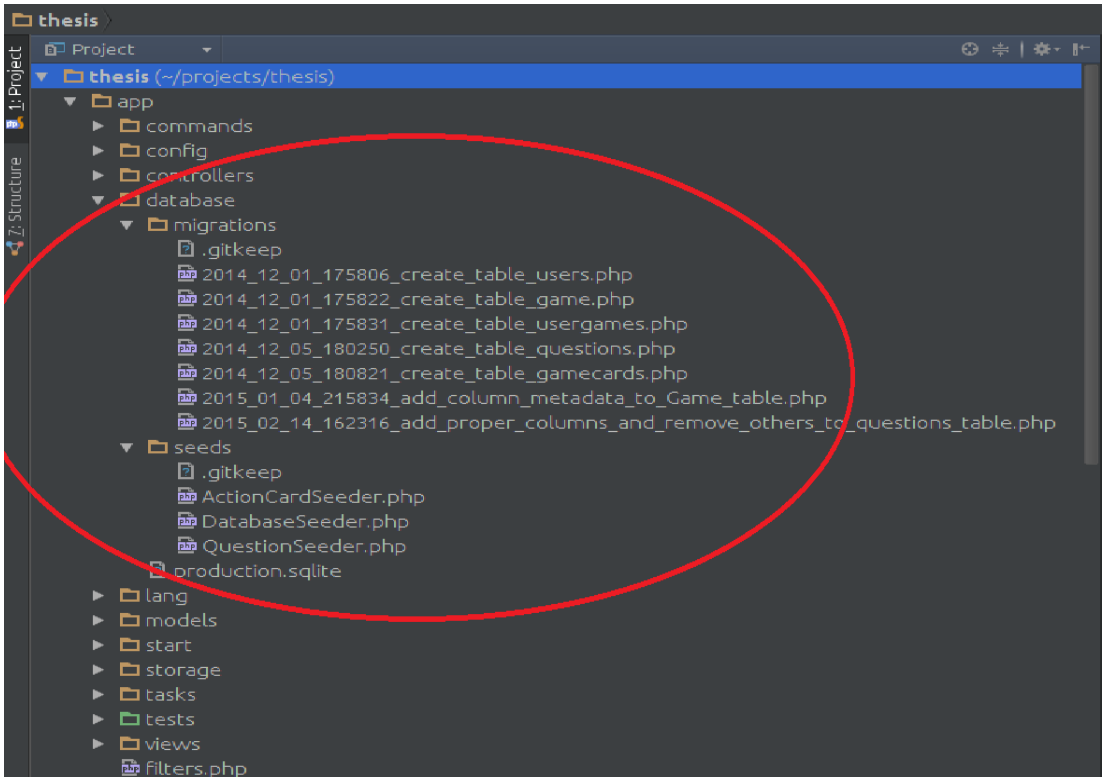
Εικόνα 36 Check Game Status

Ολοκληρώνοντας με τις βασικές τεχνολογίες, θα αναφερθούμε σε δύο τελευταίες οι οποίες είναι και οι υπεύθυνες για τον τρόπο εμφάνισης του παιχνιδιού αλλά και για την δημιουργία κάποιον εφέ πάνω σε αυτό. Η πρώτη είναι η HTML (HyperText Markup Language) στην έκδοση 5 όπως προαναφέραμε, η οποία είναι μια γλώσσα όπου δομεί και παρουσιάζει το περιεχόμενο σε όλες τις ιστοσελίδες του παγκόσμιου ιστού. Τέλος η γλώσσα που διακοσμεί το περιεχόμενο που παρουσιάζει η HTML, είναι η CSS (Cascading Style Sheets) στην έκδοση 3 η οποία παρέχει πολλές δυνατότητες και πλέον μπορεί να κάνει και απλούς μαθηματικούς υπολογισμούς. Επίσης με την τεχνολογία CSS και με ένα framework που ονομάζεται BootStrap, το οποίο έχει φτιαχτεί από την εταιρία του Twitter και μας παρέχει αρκετά έτοιμα προς χρήση αντικείμενα, έχουμε κάνει μια προσπάθεια η εφαρμογή μας να ανταποκρίνεται σε διαφορετικά μεγέθη οθονών (responsive). Αυτό σημαίνει ότι ανάλογος το μέγεθος της οθόνης μας γίνεται μια προσπάθεια να προσαρμοστεί και το μέγεθος της οθόνης του παιχνιδιού μας και των αντικειμένων που βρίσκονται πάνω σε αυτό, έτσι ώστε να είναι λειτουργικό και σε οθόνες μέχρι μεγέθους tablet. (Wikipedia, 2015) (Wikipedia, 2015) (Bootstrap)

Λοιπές Τεχνολογίες

Ολοκληρώνοντας το κεφάλαιο της υλοποίησης θα αναφερθούμε σε κάποιες τεχνολογίες οι οποίες λειτούργησαν βοηθητικά για την απλοποίηση της υλοποίηση της εργασίας μας. Αυτές οι τεχνολογίες ή παρόμοιες αυτών χρησιμοποιούνται σε όλα τα μεγάλα έργα προγραμματισμού απλοποιώντας την πολύπλοκη διαδικασία δημιουργίας εφαρμογών.

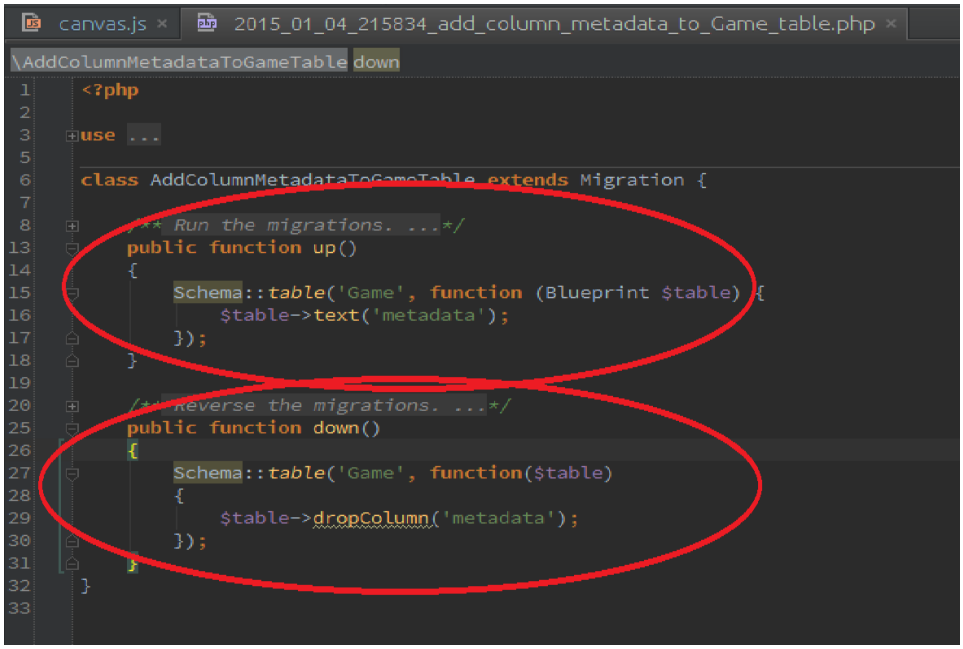
- Θα ξεκινήσουμε με τον μηχανισμό που χρησιμοποιήσαμε (migration mechanism) για την δημιουργία της βάσης δεδομένων μας αλλά και τις αλλαγές ή προσθήκες που έγιναν στην πορεία της υλοποίησης. Το framework Laravel διαθέτει ένα εργαλείο εντολών κονσόλας το οποίο ονομάζεται artisan, αυτό μας παρέχει αρκετές χρήσιμές εντολές προς χρήση για την διευκόλυνση της ανάπτυξης μιας εφαρμογής. Εκτελώντας την εντολή "php artisan list" στο τερματικό του υπολογιστή μας, εμφανίζεται μια λίστα με όλες τις διαθέσιμες εντολές. Εμείς χρησιμοποιήσαμε για αρχή την εντολή "php astisan migrate:make name_of_migration_depending_the_work_is_made_for", με το επιθυμητό όνομα κάθε φορά ανάλογος τον σκοπό δημιουργίας του. Η εντολή αυτή παράγει ένα αρχείο .php μέσα στο οποίο γραφούμε τις εντολές που θέλουμε να συντελέσουν για την δημιουργία για παράδειγμα ενός πίνακα στην βάση δεδομένων μας. Στην *εικόνα 37* βλέπουμε 7 αρχεία migration που έχουμε δημιουργήσει για τον σκοπό της εφαρμογής μας με την παραπάνω εντολή, τα 5 από αυτά συντελούν στην δημιουργία των πινάκων που αναφέραμε στην *εικόνα 27* και τα υπόλοιπα 2 είναι για αλλαγές στους πίνακες που δημιούργησαν τα προηγούμενα 5 αρχεία. Εφόσον έχουμε δημιουργήσει τα αρχεία αυτά και στην συνέχεια θέλουμε να τα εκτελέσουμε έτσι ώστε να δημιουργηθούν ή να τροποποιηθούν οι πίνακες της βάσης μας εκτελούμε την εντολή "php artisan migrate". Αυτό που κάνει η εντολή αυτή είναι να εκτελέσει σειριακά βάση της ημερομηνίας που βρίσκεται στην αρχή του ονόματος του αρχείου τα αρχεία που βλέπουμε στην *εικόνα 37*, και να δημιουργήσει ή να τροποποιήσει ότι είναι αναγκαίο στην βάση δεδομένων μας. Ολοκληρώνοντας την δημιουργία ή την τροποποίηση των πινάκων μέσω των αρχείων αυτών γράφει σε έναν πίνακα που ονομάζεται migrations και βρίσκεται στην βάση μας ότι αυτό το αρχείο έχει ήδη εκτελεστεί και οι αλλαγές του έχουν εφαρμοστεί στην βάση μας. (Laravel)



Εικόνα 37

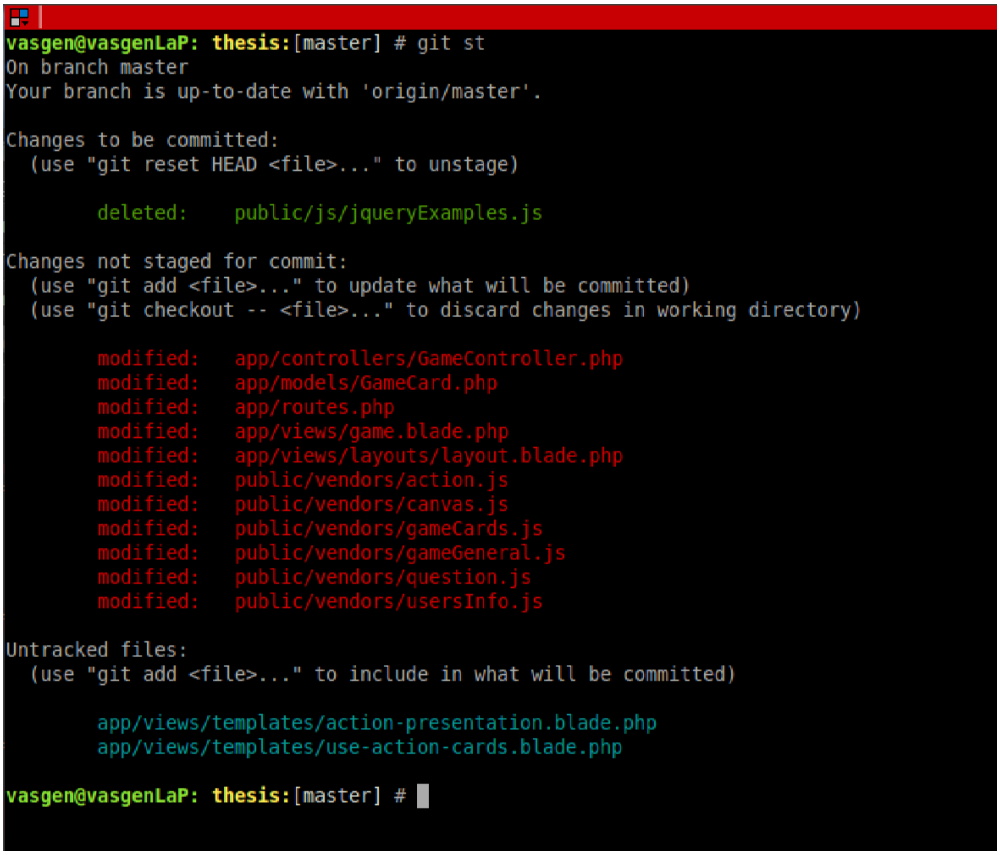
Ο λόγος που γίνεται το παραπάνω είναι γιατί πρώτον στην περίπτωση που κάποιος δημιουργήσει ένα καινούργιο αρχείο migration και θέλει να το εκτελέσει για να εφαρμοστούν οι αλλαγές του, να μην εκτελεστούν όλα τα αρχεία από την αρχή παρά μόνο αυτά τα οποία δεν έχουν περαστεί σαν εγγραφές στον πίνακα, και δεύτερον για να έχουμε την δυνατότητα ενός καταγεγραμμένου ιστορικού και έτσι όταν θα χρειαστεί να αναιρέσουμε κάποια αλλαγή από την βάση μας, με την εντολή "php artisan migrate:rollback", να μπορούμε να γνωρίζουμε την σειρά με την οποία εκτελεστήκαν οι αλλαγές. Τα αρχεία migrations μας δίνουν την δυνατότητα να αναιρέσουμε την οποία προσθήκη ή αλλαγή δημιουργούν αυτά γιατί όπως παρατηρούμε στην *εικόνα 38,* οπού βλέπουμε το εσωτερικό ενός αρχείου migration, υπάρχει μέσα σε αυτά και ο κώδικας ο οποίος αναιρεί την όποια αλλαγή ή προσθήκη έχει δημιουργεί στην βάση δεδομένων μας.

Επίσης στην *είκονα 37* παρατηρούμε κάτω από τον φάκελο migrations ότι υπάρχει ο φάκελος seeds ο οποίος περιέχει τα αρχεία με το στατικό περιεχόμενο των ερωτήσεων και των καρτών ενεργειών που προαναφέραμε. Εφόσον ολοκληρωθεί η διαδικασία των migrations και η βάση δεδομένων έχει την σωστή της δομή θα πρέπει να εκτελέσουμε την εντολή "php artisan db:seed" έτσι ώστε το περιεχόμενο των seed αρχείων μας να περαστεί στην βάση μας και να μπορέσουμε να το χρησιμοποιήσουμε τις πληροφορίες που παρέχουν στους πίνακες για τις ανάγκες του παιχνιδιού μας.



Εικόνα 38

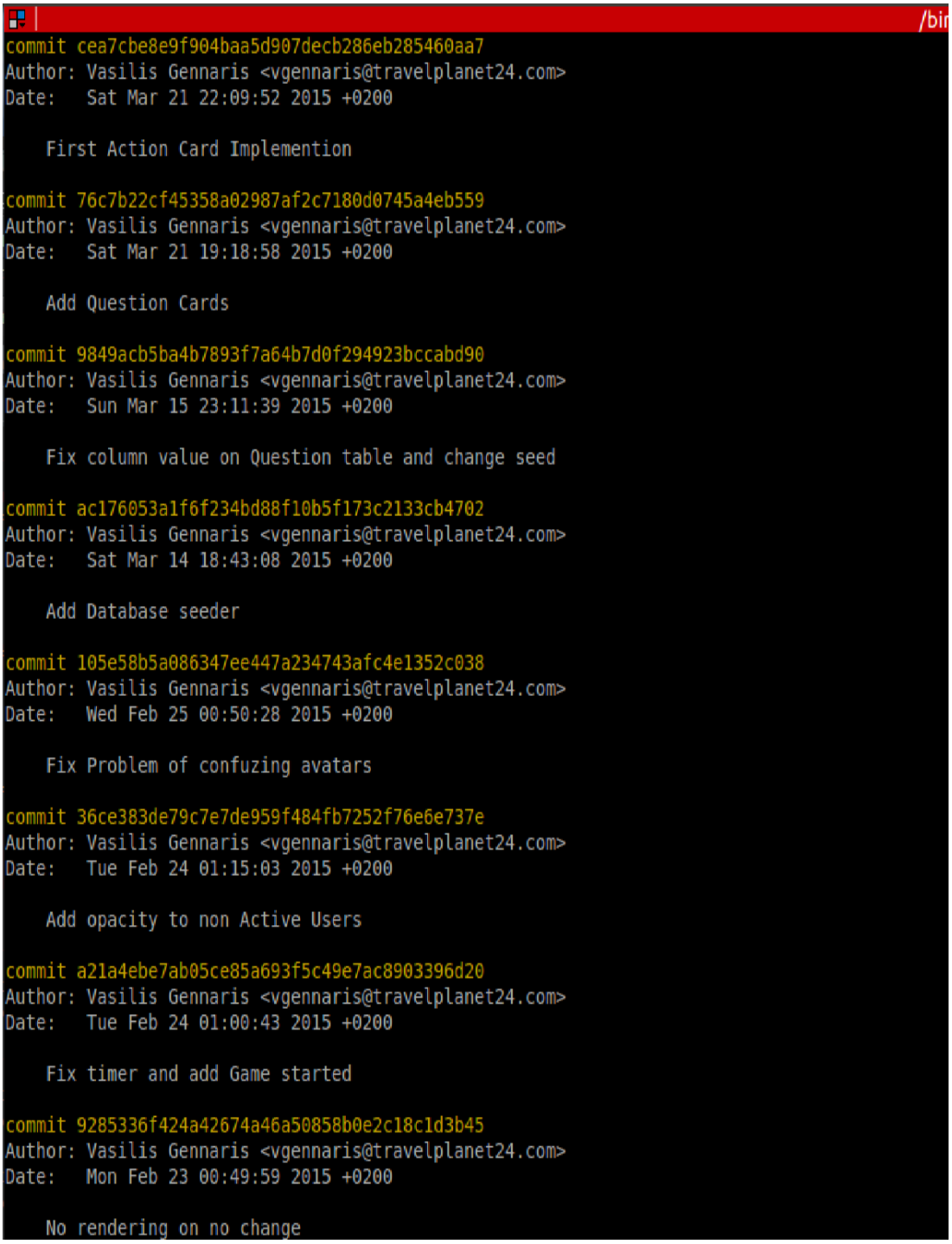
- Συνεχίζοντας επίσης χρησιμοποιήθηκαν συνδυαστικά η υπηρεσία Bitbucket με το λογισμικό Git. Το Bitbucket είναι μια πλατφόρμα στην οποία μπορεί κανείς να δημιουργήσει έναν λογαριασμό και να αποθηκεύσει τον κώδικα της εφαρμογής στην οποία εργάζεται. Η χρήση της πλατφόρμας αυτής είναι δωρεάν για ντετερμινιστικό αριθμό εφαρμογών τις οποίες μπορεί να φιλοξενήσει σε κάθε λογαριασμό. Σε συνδυασμό με το λογισμικό ανοικτού κώδικα Git, το οποίο είναι εργαλείο έλεγχου της έκδοσης του κώδικά και των αλλαγών αυτού, δημιουργείται ένα πολύ δυνατό συνδυασμός που διευκολύνει την δουλεία των προγραμματιστών ειδικά όταν εργάζονται πολλά άτομα πάνω σε ένα έργο ή όταν η εργασία γίνεται απομακρυσμένα, κάνοντας την διαδικασία των αλλαγών πιο συντονισμένη και εύκολα ανιχνεύσιμη. Επίσης ένα πολύ σημαντικό χαρακτηριστικό του Git είναι ότι γράφοντας κώδικά στον IDE μας, αυτό αυτομάτως αναγνωρίζει τις αλλαγές που υπάρχουν σε σύγκριση με τον κώδικά της εφαρμογής μας που βρίσκεται αποθηκευμένος στο Bitbucket απλά εκτελώντας την εντολή “git status” στο τερματικό του υπολογιστή μας. Όπως παρατηρούμε στην *εικόνα 39* εκτελώντας την εντολή αυτή μας εμφανίζονται διάφορα αρχεία με διαφορετικά χρώματα το καθένα. Παρατηρούμε λοιπόν ότι έχουμε ένα διαγραμμένο αρχείο με χρώμα πράσινο, έντεκα αρχεία με κόκκινο χρώμα τα οποία έχουν υποστεί αλλαγές σε σχέση με τον κώδικα που βρίσκεται στο Bitbucket, και τέλος δύο νέα αρχεία τα οποία εμφανίζονται με μπλε χρώμα.



Εικόνα 39

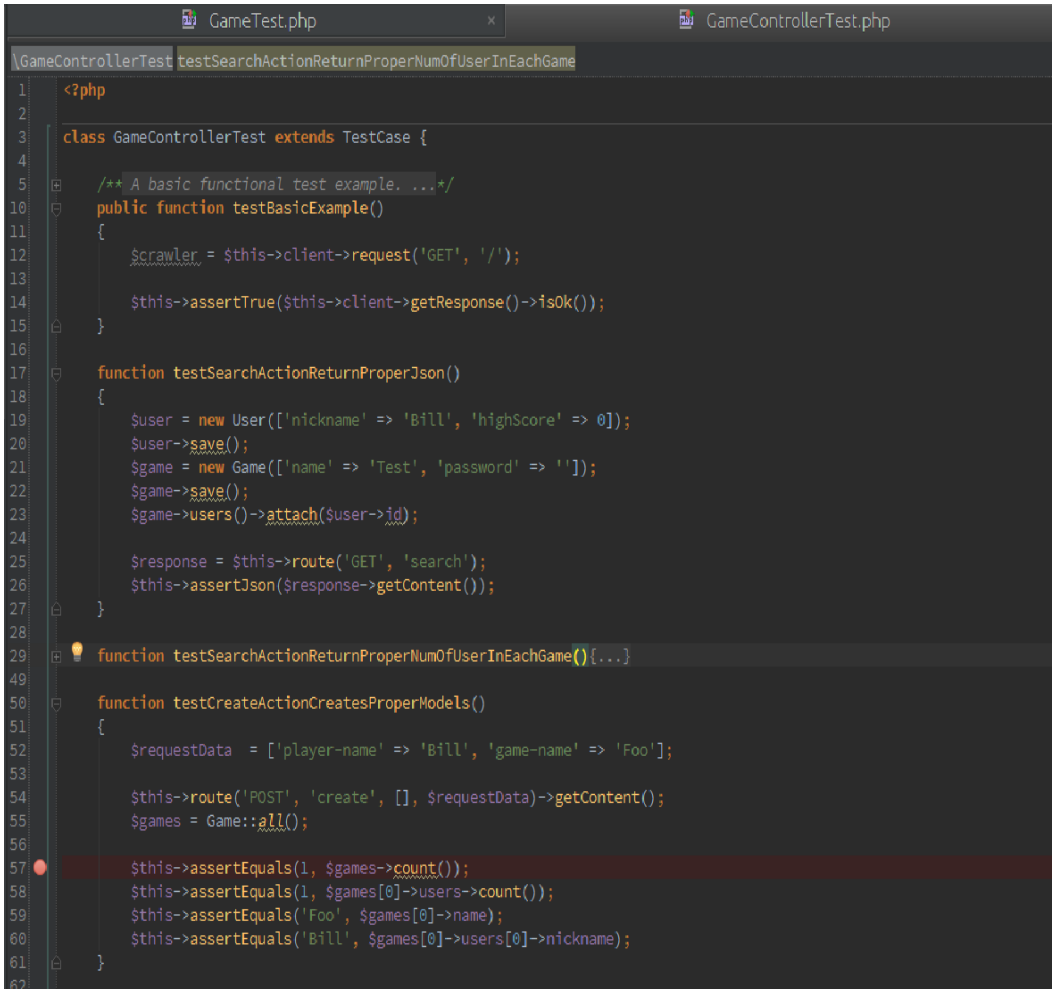
Αν οι αλλαγές αυτές είναι ολοκληρωμένες και θέλουμε να τις προωθήσουμε, στην "αποθήκη" του κώδικα μας στο Bitbucket, τότε με μια αλληλουχία εντολών του git, "git add", "git commit –m commit_name" και "git push", μπορούμε να τις στείλουμε στο σημείο που είναι αποθηκευμένη η εφαρμογή μας στο Bitbucket σαν ένα πακέτο κώδικά με αλλαγές οι οποίες χαρακτηρίζονται μοναδικά από μια αλληλουχία αριθμών (sha1). Εν συνεχεία εκτελώντας στο τερματικό μας την εντολή "git log" μπορούμε να δούμε όλη την αλληλουχία από πακέτα αλλαγών που έχουν προωθηθεί στο απομακρυσμένο σημείο αποθήκευσης της εφαρμογής μας είτε από εμάς είτε από οποιονδήποτε άλλο εργάζεται στην εφαρμογή μας. Αυτό παρατηρούμε στην *εικόνα 40* οπού βλέπουμε ένα στιγμιότυπο από το τερματικό μας έχοντας εκτελέσει την εντολή "git log", παρατηρούμε τα πακέτα με τις αλλαγές του κώδικα ταξινομημένα με χρονολογική σειρά από το νεότερο πιο ψηλά μέχρι το παλαιότερο χαμηλότερα. Το λογισμικό Git έχει πάρα πολλές δυνατότητες που "λύνουν" τα χεριά των προγραμματισμών. Στην δίκια μας εφαρμογή μας βοήθησε πάρα πολύ επειδή η εργασία υλοποιήθηκε από διαφορά σημεία, ώστε να μην χρειάζεται να μεταφέρουμε κάθε φορά τον φορητό μας υπολογιστή, γιατί ότι γράφαμε σε κώδικα και ήταν ολοκληρωμένο απευθείας αυτό το προωθούσαμε στον λογαριασμό μας στο Bitbucket έτσι ώστε να μπορούμε να έχουμε πρόσβαση σε αυτό από κάποιον άλλο υπολογιστή και να συνεχίσουμε την εργασία μας εκεί. Επίσης αν συνειδητοποιούσαμε ότι δημιουργήθηκε κάποιο πρόβλημα και το οποίο προήλθε από το τελευταίο τμήμα

κώδικα που προωθήσαμε στο Bitbucket, και δεν το είχαμε προβλέψει πριν γίνει η αποστολή του αυτού, τότε το Git μας δίνει την δυνατότητα να κάνουμε αναίρεση του τελευταίου κομματιού ή όσων κομματιών είναι απαραίτητο με σκοπό να αφαιρέσουμε το πρόβλημα και να επανελέγξουμε τον κώδικα, χρησιμοποιώντας την εντολή "git revert sha1 " και εν συνεχεία "git push". Αυτή η διαδικασία αντιστρέφει τον κώδικα με κωδικό sha1 στον κώδικα της εφαρμογή μας που βρίσκεται στο BitBucket. (Git) (Wikipedia) (Bitbucket)



Εικόνα 40

- Μια ακόμη τεχνολογία που χρησιμοποιήθηκε και δεν έχει άμεση σχέση με την υλοποίηση της εφαρμογής του παιχνιδιού αυτού καθ αυτού, είναι το PHPUnit το οποίο είναι ένα framework όπου χρησιμοποιείται για τον έλεγχο (testing) του κώδικα που έχει ήδη γραφτεί. Έχει δημιουργηθεί σε PHP γλώσσα και εφαρμόζεται σε κάθε τμήμα κώδικά (function) περνώντας σε αυτό έτοιμα δεδομένα και περιμένοντας μια συγκεκριμένη απόκριση από αυτόν. Έτσι γνωρίζοντας το τελικό αποτέλεσμα αλλά και τα δεδομένα που χρειάζονται για να δημιουργηθεί αυτό, θέτουμε τα δεδομένα στην function σαν είσοδο και ελέγχουμε ότι με την ολοκλήρωσή της θα φέρει το παραπάνω επιθυμητό αποτέλεσμα. Ο έλεγχος του κώδικά με αυτόν τον τρόπο γίνεται για τον λόγο του ότι μια αλλαγή που κάνουμε στον κώδικά μας μπορεί να επηρεάζει κάποιο σημείο το οποίο δεν έχουμε προβλέψει, έτσι πριν προωθήσουμε τις αλλαγές μας στο Bitbucket θα πρέπει να ελέγχουμε ότι όλα μας τα τεστ λειτουργούν σωστά και σαν συμπέρασμα αυτού ότι ο νέος μας κώδικάς μας δεν επηρέασε κάτι στο υπόλοιπο σύνολο. Αυτή η διαδικασία προσθέτει στην εφαρμογή μας ένα ακόμη επίπεδο ασφάλειας και ποιότητας του κώδικά που γράφεται. Αυτό είναι μια διαδικασία που είναι καλό να ακολουθείτε σε όλα τα έργα και εφαρμογές και είναι μέρος μιας πιο γενικής τεχνικής που ονομάζεται “Continuous Integration” (CI). Στην *εικόνα 41* βλέπουμε ένα αρχείο έλεγχου του βασικού μας κώδικά. Στην γραμμή από 50 έως 61 βλέπουμε ότι δημιουργούμε ένα παιχνίδι στην βάση μας και μετά κάνουμε έλεγχο αν όντως υπάρχει ένα παιχνίδι και ένας χρήστης στην βάση δεδομένων μας, όπως επίσης ελέγχουμε, στις γραμμές 59 και 60, ότι τα ονόματα των παραπάνω είναι σωστά. (PHPunit)



Εικόνα 41

Ολοκληρώνοντας με τις τεχνολογίες, χρησιμοποιήσαμε ένα αρχείο που ονομάζεται Makefile και μέσα σε αυτό με καθορισμένη γραφή μπορούμε να γράψουμε κάποιες εντολές τις οποίες θα εκτελούμε από το τερματικό μας μηχάνημα βάζοντας το όνομα make ακολουθούμενο από τον τίτλο της εντολής. Για παράδειγμα έχουμε φτιάξει μια εντολή που λέγεται test μέσα στο Makefile αρχείο. Η εντολή είναι η εξής:

```
test:
	@echo "\nStarting phpUnit...\n"
	vendor/bin/phpunit
```

Αυτή κατά την εκτέλεση της, μέσω της εντολής “make test”, αυτό που κάνει είναι να εμφανίζει το μήνυμα “Starting phpUnit...” στην οθόνη μας και στην συνέχεια εκτελεί όλα τα τεστ που έχουν γραφτεί για την εφαρμογή. Αν τα τεστ ολοκληρωθούν χωρίς λάθη τότε εμφανίζει ένα μήνυμα “OK” αλλιώς μας γράφει “Fail” και μας αναφέρει πιο τεστ είχε λάθος ώστε να εντοπίσουμε το λάθος και να το διορθώσουμε. (Wikipedia, 2015)

Μελλοντικά Σχέδια

Κάθε εφαρμογή που υπάρχει ή δημιουργείται από την αρχή έχει την δυνατότητα να εξελίσσεται από τον δημιουργό της και τεχνολογικά αλλά και όσον αφορά το παρουσιαστικό κομμάτι αέναος. Ειδικά οι διαδικτυακές εφαρμογές, οι τεχνολογίες των οποίων εξελίσσονται και προχωρούν με πολύ γρήγορους ρυθμούς, θα πρέπει να ακολουθούν τις αλλαγές αυτές αν θέλουν οι δημιουργοί οι εφαρμογές τους να παραμένουν ανταγωνίστηκες σε έναν τόσο γρήγορα αναπτυσσόμενο χώρο.
Στο κεφάλαιο αυτό θα αναφέρουμε μελλοντικά σχέδια και βελτιώσεις τις οποίες θα μπορούσαμε να εφαρμόσουμε στο παιχνίδι μας.

Βελτιώσεις στο Σενάριο του Παιχνιδιού

Θα ξεκινήσουμε με τις βελτιώσεις που θα μπορούσαμε να κάνουμε στο υπάρχων σενάριο του παιχνιδιού.

- Μπορούμε να υλοποιήσουμε τη δυνατότητα αντί ένας παίκτης να είναι αναγκασμένος να παίξει το παιχνίδι με κάποιον άλλο ο οποίος πρέπει να είναι φυσικό πρόσωπο να υπάρχει η δυνατότητα να παίζει με αντίπαλο έναν παίκτη τον οποίο θα διαχειρίζεται ο υπολογιστής. Με αυτήν την λειτουργία θα καταφέρουμε να έχουμε περισσότερο κοινό το οποίο θα παίξει το παιχνίδι μας λόγω της απαλοιφής της ανάγκης για εύρεση το λιγότερο ενός ακόμη παίκτη.
- Επίσης θα μπορούσαμε τις ήδη υπάρχουσες κάρτες ενεργειών να μπορεί ένας χρήστης να τις χρησιμοποιεί ενάντιων ενός άλλου παίκτη. Δηλαδή θα μπορούσαν να υπάρχουν κάρτες που για παράδειγμα ο χρήστης Α δεν θα επέτρεπε στον χρήστη Β να ακολουθήσει την σκάλα όταν η ζαριά του θα τελείωνε σε κάποιο τετράγωνο από όπου θα ξεκινούσε αυτή ή θα μπορούσε να αναγκάζει κάποιον αντίπαλο παίκτη να κινηθεί κάποια τετράγωνα πίσω στο ταμπλό ώστε να περάσει αυτός μπροστά. Αυτό σαν λειτουργία θα δημιουργούσε την ανάγκη για περαιτέρω στρατηγική σκέψη μεταξύ τον παικτών.
- Ένα ακόμη σημείο που επίσης έχει να κάνει με το σενάριο του παιχνιδιού, θα ήταν οι παίκτες να μην ξεκινούν το παιχνίδι με την σειρά την οποία εισήχθηκαν σε αυτό αλλά μέσω του αποτελέσματος από την ρίψη ζαριού και να ξεκινούσαν με σειρά από τον μεγαλύτερο αριθμό ρίψης προς τον μικρότερο. Αυτό θα μας έδινε την δυνατότητα να μπορούμε αναλόγως με την σειρά τερματισμού να δίνουμε στους παίκτες κάποιους βαθμούς οι οποίοι θα προστίθεντο στην τελική βαθμολογία του κάθε παίκτη αναλόγως την σειρά τερματισμού τους. Για παράδειγμα ο πρώτος να παίρνει 4 πόντους, ο δεύτερος 3, ο τρίτος 2 και ο τελευταίος 1. Σαν αποτέλεσμα αυτό θα δημιουργούσε στους παίκτες την ανάγκη να δημιουργούν έναν καλύτερο σχεδιασμό στρατηγικής για τον τρόπο κίνησης τους πάνω στο ταμπλό του παιχνιδιού με σκοπό τον τερματισμό τους με την μεγαλύτερη δυνατή βαθμολογία.
- Συνεχίζοντας θα μπορούσαμε να είχαμε μεγαλύτερο εύρος και πεδίο ερωτήσεων ώστε

να δίνεται η δυνατότητα και σε χρήστες που δεν έχουν γνώσεις πληροφορικής να μπορούν να συμμετέχουν και να γνωρίζουν να απαντούν στις ερωτήσεις του παιχνιδιού. Ο λόγος όπως προαναφέραμε που δεν έγινε κάτι τέτοιο είναι ότι για τις ανάγκες τις εργασίας μας έπρεπε να έχουμε μικρό εύρος και πεδίο για να μπορέσουμε να δημιουργήσουμε το παιχνίδι στον χρόνο τον οποίο διαθέταμε και ταυτόχρονα αυτό να είναι λειτουργικό.

- Επίσης στην οθόνη του παιχνιδιού εφόσον μιλάμε για ένα διαδικτυακό παιχνίδι θα μπορούσε να υπάρχει ένα πεδίο ανταλλαγής μηνυμάτων μεταξύ των παικτών ώστε να μπορούν να έχουν κάποια υποτυπώδη επικοινωνία μεταξύ τους.
- Τελειώνοντας, ένα σημαντικό σημείο το οποίο θα μπορούσε να βελτιωθεί είναι αυτό της καλύτερης ανταμοιβής των παικτών. Για παράδειγμα εκτός από το σημείο στον τερματισμό στο οποίο εμφανίζεται μια εικόνα με το όνομα, το σκορ του χρήστη και ένα μήνυμα ότι είναι ο νικητής του παιχνιδιού θα μπορούσε να εμφανίζεται σε κάποιο σημείο των οθονών του παιχνιδιού τα ονόματα των παικτών με τις καλύτερες βαθμολογίες. Επίσης στο σημείο της οθόνης του υπολογιστή που εμφανίζονται οι πληροφορίες των χρηστών θα μπορούσαμε να έχουμε κάποια αστέρια αναλόγως τις συνολικές νίκες που έχει κάνει ο παίκτης αυτός.

Βελτιώσεις στο Σχεδιαστικό Μέρος του Παιχνιδιού

- Η τεχνολογία του διαδικτύου και των εφαρμογών, όσον αφορά το σχεδιαστικό, έχουν την τάση να δημιουργούνται στις μέρες μας για διαφορά μεγέθη οθονών, όπως κινητά και tablet μέχρι οθόνες 27 ιντσών ή και ακόμη μεγαλύτερες. Για τον λόγο αυτό έχει γίνει κάποια προσπάθεια ώστε το παιχνίδι μας να είναι λειτουργικό σε διαφόρων μεγεθών οθόνες, δηλαδή να ανταποκρίνεται σε εναλλαγές του μεγέθους της οθόνης. Παρόλα αυτά σε μια οθόνη κινητού το παιχνίδι μας δεν είναι λειτουργικό. Οπότε μια μελλοντική βελτίωση σε αυτήν την κατεύθυνση θα ήταν κάτι το οποίο θα βοηθούσε να γίνει το παιχνίδι μας προσιτό σε μεγαλύτερο εύρος κοινού όπου θα μπορούσε οποιαδήποτε στιγμή να παίξει με αυτό.
- Επίσης στην διαδικασία ρίψης του ζαριού θα μπορούσε να προστεθεί ένα εφέ το οποίο θα έκανε το ζάρι να φαίνεται ότι εναλλάσσεται μεταξύ των αριθμών μέχρι να καταλήξει στον τελικό, αντί αυτού που συμβαίνει τώρα όπου δείχνει απευθείας τον τελικό αριθμό. Με απλά λόγια να προσπαθήσουμε να προσομοιάσουμε το πραγματικό ζάρι ενός επιτραπέζιου.
- Τελευταίο και πολύ σημαντικό είναι η χρήση ενός μουσικού χαλιού στο παιχνίδι μας το οποίο κάνει πιο ευχάριστη την παραμονή και την όλη διαδικασία του παιχνιδιού στον χρήστη. Επίσης θα μπορούσαν να χρησιμοποιηθούν κάποιοι ήχοι προσομοίωσης κινήσεων ή επιβράβευσης, όπως για παράδειγμα ήχος στην ρίψη του ζαριού ή στην κίνηση του παίκτη πάνω στο ταμπλό.

Βελτιώσεις στο Τεχνολογικό Μέρος του Παιχνιδιού

- Στο τεχνολογικό τμήμα του παιχνιδιού, θα μπορούσαμε μελλοντικά να χρησιμοποιήσουμε web-sockets για το κομμάτι της επικοινωνίας μεταξύ client – server αλλά και για το κομμάτι επικοινωνίας client – client δηλαδή τον συγχρονισμό των δεδομένων μεταξύ των παικτών ενός παιχνιδιού. Η τεχνολογία που έχει χρησιμοποιηθεί για τον σκοπό αυτόν στο παιχνίδι μας είναι το AJAX (Asynchronous JavaScript And XML) στην οποία αναφερθήκαμε και προηγούμενος. Ανά τακτά χρονικά διαστήματα ο client στέλνει ένα request στον server και ρωτάει για αλλαγές που υπάρχουν στο παιχνίδι από άλλους παίκτες ώστε να ενημερώσει τα δεδομένα του, επίσης στέλνεται κάποιο request από τον client όταν αλλάξει κάτι στην κατάσταση του παιχνιδιού από τον παίκτη ώστε να ενημερωθούν και οι υπόλοιποι. Αυτά τα request λειτουργούν μέσω του πρωτοκόλλου HTTP το όποιο σε κάθε αποστολή ανοίγει μια συνεδρία με τον server η οποία κλείνει όταν τελειώσει η επικοινωνία με αυτόν. Αυτό έχει σαν αποτέλεσμα να στέλνονται request ακόμη και αν δεν υπάρχει κάποια αλλαγή στα δεδομένα του παιχνιδιού λόγο της αδυναμίας των clients να γνωρίζουν αν υπάρχουν αλλαγές στο παιχνίδι. Επίσης επειδή σε κάθε request πρέπει να δημιουργηθεί μια καινούργια σύνδεση στέλνονται και πληροφορίες για την δημιουργία αυτής, αυτό δημιουργεί μεγαλύτερη διακίνηση δεδομένων. Τα προηγούμενα θα αποτελέσουν πρόβλημα αν το παιχνίδι μας αποκτήσει πολύ μεγάλη επισκεψιμότητα και η λύση σε αυτό θα μπορούσε να είναι η τεχνολογία web sockets. Η διαφορά στην τεχνολογία αυτή είναι ότι καθ' όλη την διάρκεια του παιχνιδιού υπάρχει μια ανοιχτή επικοινωνία μεταξύ client – server η οποία δημιουργείτε μέσω ενός request στην αρχή του παιχνιδιού μέσω TCP πρωτοκόλλου και δεν χρειάζεται κάθε φορά η δημιουργία νέα σύνδεσης με τον server για την επικοινωνία. Με αυτό τον τρόπο μειώνεται η επιπρόσθετη πληροφορία που στέλνεται μέσω του HTTP request για την έναρξη της συνεδρίας με τον browser με αποτέλεσμα να έχουμε πιο απλή και γρήγορη επικοινωνία μεταξύ client – server. Επίσης ο κάθε client θα δέχεται την νέα πληροφορία του παιχνιδιού αν αλλάξει κάτι σε αυτό από τον οποιοδήποτε παίκτη απευθείας χωρίς να χρειάζεται να στέλνει συνέχεια requests για να ρωτάει για τυχόν αλλαγές.

Βιβλιογραφία

Abode Illustrator. (n.d.). Retrieved 03 25, 2015, from Abode Illustrator: http://www.adobe.com/products/illustrator.html

Adobe Photoshop. (n.d.). Retrieved 03 25, 2015, from Adobe Photoshop: http://www.adobe.com/gr_en/products/photoshop.html

Apache. (n.d.). Retrieved 03 24, 2015, from Apache: https://www.apache.org/

Balsamiq. (n.d.). Retrieved 03 26, 2015, from Balsamiq: https://balsamiq.com/

Bitbucket. (n.d.). Retrieved 03 26, 2015, from Bitbucket: https://bitbucket.org/

Bootstrap. (n.d.). Retrieved 03 25, 2015, from Bootstrap: http://getbootstrap.com/

Canvas. (n.d.). Retrieved 03 20, 2015, from Weschools: http://www.w3schools.com/html/html5_canvas.asp

Git. (n.d.). Retrieved 03 25, 2015, from Git: http://git-scm.com/

Jcanvas. (n.d.). Retrieved 03 26, 2015, from Jcanvas: http://calebevans.me/projects/jcanvas/

jQuery. (n.d.). Retrieved 03 20, 2015, from jQuery: https://jquery.com/

Laravel. (n.d.). Retrieved 03 25, 2015, from Artisan: http://laravel.com/docs/5.0/artisan

MySql. (n.d.). Retrieved 03 25, 2015, from MySql: http://www.mysql.com/

PHP. (n.d.). Retrieved 03 24, 2015, from PHP: http://php.net/

PHPunit. (n.d.). Retrieved 03 27, 2015, from PHPunit: https://phpunit.de/

Schell, J. (2008). *The Art of Game Design.* Mogran Kaufmann.

W3schools. (2015, 03 15). Retrieved 03 25, 2015, from w3schools: http://www.w3schools.com/ajax/

WebSockets. (n.d.). Retrieved 03 22, 2015, from Mozilla Developers: https://developer.mozilla.org/en-US/docs/WebSockets

Wikipedia. (n.d.). Retrieved 03 25, 2015, from Ajax: http://en.wikipedia.org/wiki/Ajax_%28programming%29

Wikipedia. (n.d.). Retrieved 03 25, 2015, from Git: http://en.wikipedia.org/wiki/Git_%28software%29

Wikipedia. (2015, 03 25). Retrieved 03 26, 2015, from

http://en.wikipedia.org/wiki/Snakes_and_Ladders

Wikipedia. (2015, 03 24). Retrieved 03 26, 2015, from Wikipedia: http://en.wikipedia.org/wiki/SQL

Wikipedia. (2015, 02 18). Retrieved 03 25, 2015, from Wikipedia: http://en.wikipedia.org/wiki/Apache_HTTP_Server

Wikipedia. (2015, 03 26). Retrieved 03 27, 2015, from Wikipedia: http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

Wikipedia. (2015, 03 20). Retrieved 03 26, 2015, from Wikipedia: http://en.wikipedia.org/wiki/Integrated_development_environment

Wikipedia. (2015, 03 18). Retrieved 03 26, 2015, from Wikipedia: http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Wikipedia. (2015, 03 30). Retrieved 03 30, 2015, from Wikipedia: http://en.wikipedia.org/wiki/Uniform_resource_identifier

Wikipedia. (2015, 03 20). Retrieved 03 26, 2015, from Wikipedia: http://en.wikipedia.org/wiki/WebSocket

Wikipedia. (2015, 03 20). Retrieved 03 25, 2015, from HTML5: http://en.wikipedia.org/wiki/HTML5

Wikipedia. (2015, 03 15). Retrieved 03 25, 2015, from CSS: http://en.wikipedia.org/wiki/Cascading_Style_Sheets

Wikipedia. (2015, 03 10). Retrieved 03 26, 2015, from Makefile: http://en.wikipedia.org/wiki/Makefile

Wikipwdia. (2015, 05 12). Retrieved 03 25, 2015, from Wikipedia: http://en.wikipedia.org/wiki/Game_design

Παράρτημα Α – Οδηγός χρήση και εγκατάστασης

Για την σωστή εγκατάσταση της εφαρμογής του παιχνιδιού μας σε κάποιον server χρειάζονται να εκτελεστούν οι σειριακά οι παρακάτω ενέργειες.

- Εγκατάσταση του λογισμικού Apache.
- Εγκατάσταση της γλώσσας PHP και του απαιτήσεων που έχει αυτή.
- Εγκατάσταση της sql για την δημιουργία και διαχείριση της βάσης δεδομένων.
- Εγκατάσταση της MySql γλώσσας επικοινωνίας βάσεων δεδομένων.
- Μετά την σύνδεση στην mysql με την εντολή "mysql –u root –p root" εκτελούμε την εντολή "Create Database geakysnake;" για την δημιουργία της βάσης στον υπολογιστή εγκατάστασης.
- Στην συνέχεια αντιγραφούμε σε κάποιο σημείο του υπολογιστή μας τον πηγαίο κώδικα.
- Εφόσον έχουμε στον υπολογιστή μας τον κώδικα, πηγαίνουμε στο αρχείο hosts του υπολογιστή μας και θέτουμε την IP και το URI της εφαρμογής μας. Πχ 127.0.0.1 geekysnake.local
- Μεταβαίνουμε στο φάκελο apache2/sites-available και φτιάχνουμε ένα .conf αρχείο το οποίο θα περιέχει τις απαραίτητες πληροφορίες για το που βρίσκεται το κεντρικό σημείο από το οποίο ξεκινά η εφαρμογή μας.

```
<VirtualHost *:80>
        ServerName geekysnake.local
        DocumentRoot /home/project/geekysnake/public
</VirtualHost>
```

- Εκτελούμε την εντολή composer install για να εγκαταστήσουμε όλες τις απαραίτητες βιβλιοθήκες που χρησιμοποίει η εφαρμογή μας
- Εκτελούμε σειριακά τις εντολές "php artisan migrate" και "php artisan db:seed" για να δημιουργήσουμε το σχήμα της βάσης και να γεμίσουμε τους πίνακες με τις απαραίτητες πληροφορίες
- Τέλος πατώντας τρέχοντας την εντολή "sudo service apache2 restart" και εν συνεχεία στον browser μας ζητώντας την διεύθυνση geekysnake.local θα πρέπει να δούμε την πρώτη οθόνη του παιχνιδιού μας.

Παράρτημα Β – Κώδικας

GameController.php

```
<?php

class GameController extends BaseController {

      function searchAction()
      {
        $games = Game::orderBy('created_at', 'DESC')->get();
        foreach($games as $game) {
            $metadata = json_decode($game->metadata);
            if (time() - $metadata->gameStats->gameStartedTime < 10800) {
                $game->usersNum = $game->countUsers();
                $game->gameStarted = $metadata->gameStats->gameStarted;
            } else {
                $game->delete();
            }
        }
        if(Request::ajax()) {
            return $games->toJson();
        }

        return View::make('search')->with('games', $games->toJson());
    }

      function createAction()
      {
        if (!Session::has('user')) { //Fix data that gets if session hasn't user.
            $user = new User(['nickname' => Input::get('player-name', 'User'), 'highScore' =>
0]);
            $user->save();
            $user->addUserToSession();
        } else {
            $user = Session::get('user');
        }

        $gameStartedTime = time();

        $metadata = ['users'=>[
                            $user->id => [
                                'name' => $user->nickname,
                                'score' => 0,
                                'currentPosition' => 0,
                                'cards' => 0,
                                'activeUser' => true,
                                'coordinates' => ['x' => 60, 'y'=> 456],
                                "timeAdded" => $gameStartedTime,
                                "actionCards" => []
                            ]
                        ],
                    "gameStats" => [
                        "numPlayers" => 1,
                        "gameFinished" => false,
                        "gameStartedTime" => $gameStartedTime,
                        "gameStarted" => false,
                        "questionCardsPlayed" => []
                        ]
                    ];
        $isPrivate = Input::get('group-private') == 'private';
```

```
        $game = new Game(['name' => Input::get('game-name', 'Game'), 'isPrivate' => $isPrivate
? 1 : 0,
                    'password' => $isPrivate ? sha1(Input::get('password')) : ''
                ]);
        $game->save();
        $game->addGameToSession();
        $metadata['gameID'] = $game->id;
        $game->metadata = json_encode($metadata);
        $game->save();
        $game->users()->attach($user->id);

        return Redirect::route('game', ['id' => $game->id]);
    }

    function addPlayerToGameAction($id)
    {
        $password = Input::get('new-player-join-password');
        $game = Game::find($id);
        $game->addGameToSession();

        if ($password != '') {
            if ($game->password != sha1($password)) {
                return Redirect::back()->with('message','Wrong Password');
            }
        }

        if (!Session::has('user')) {
            $user = new User(['nickname' => Input::get('player-name', 'User'), 'highScore' =>
0]);
            $user->save();
            $user->addUserToSession();
        } else {
            $user = Session::get('user');
        }

        $metadata = json_decode($game->metadata);
        $userToBeAdded = new stdClass();
        $userToBeAdded->name = $user->nickname;
        $userToBeAdded->score = 0;
        $userToBeAdded->currentPosition = 0;
        $userToBeAdded->cards = 0;
        $userToBeAdded->activeUser = false;
        $userToBeAdded->coordinates = ['x' => 60, 'y'=> 456];
        $userToBeAdded->timeAdded = time();
        $userToBeAdded->actionCards = [];

        $userID = $user->id;
        $metadata->users->$userID = $userToBeAdded;
        $metadata->gameStats->numPlayers += 1;
        $game->metadata = json_encode($metadata);
        $game->save();

        $userAlreadyAttachedToThisGame = $game->users->contains($user->id);

        if(!$userAlreadyAttachedToThisGame) {
            $game->users()->attach($user->id);
        }

        return Redirect::route('game', ['id' => $game->id]);
    }

    function joinAction($id)
    {
        $user = Session::get('user');
        $game = Game::find($id);
        $game->addGameToSession();
```

```
        $metadata = json_decode($game->metadata);
        $userToBeAdded = new stdClass();
        $userToBeAdded->name = $user->nickname;
        $userToBeAdded->score = 0;
        $userToBeAdded->currentPosition = 0;
        $userToBeAdded->cards = 0;
        $userToBeAdded->activeUser = false;
        $userToBeAdded->coordinates = ['x' => 60, 'y'=> 456];
        $userToBeAdded->timeAdded = time();
        $userToBeAdded->actionCards = [];

        $userID = $user->id;
        $metadata->users->$userID = $userToBeAdded;
        $metadata->gameStats->numPlayers += 1;
        $game->metadata = json_encode($metadata);
        $game->save();

        $userAlreadyAttachedToThisGame = $game->users->contains($user->id);

        if(!$userAlreadyAttachedToThisGame) {
            $game->users()->attach($user->id);
        }

        return Redirect::route('game', ['id' => $game->id]);
    }

    function playAction($id)
      {
        $currentGame = Game::findOrFail($id);

        if (Session::has('user')) {
            return View::make('game')->with(['players' => $currentGame->users->toJson(),
'game' => $currentGame, 'boardData' => json_encode(Game::$squares), 'actionCards' =>
json_encode(GameCard::$actionCards), 'userIDFromSession' => Session::get('user')->id]);
        } else {
            return Redirect::to('search')->withErrors(['error' => 'The game not exist']);
        }
    }

    function leaveGameAction($gameID)
    {
        $user = Session::get('user');

        $game = Game::find($gameID);
        $game->users()->detach($user->id);

        $gameMetadata = json_decode($game->metadata);
        foreach($gameMetadata->users as $key => $metadataUser) {
            if ($key == $user->id) {
                unset($gameMetadata->users->{$key});
            }
        }
        $game->metadata = json_encode($gameMetadata);
        $game->save();

        return Redirect::to('search')->with(['msg' => 'You left game successfully']);
    }

    function updateMetadataAction($id)
    {
        $userKeys = [];
        $i = 0;
        $playersFinish = 0;
        $game = Game::find($id);
        $user = Session::get('user');
```

```
        $blocksToBeAdded = Input::get('position');
        $pointFromActionCard = Input::get('actionCardValue');
        $activeUser = Input::get('activeUser');
        $gameStarted = Input::get('gameStarted');
        $jumpTo = (int)Input::get('jumpTo');


        do {
            $game = Game::find($id);
            sleep(0.2);
        } while($game->lock);

        $game->lock = true;
        $game->save();

        $gameMetadata = json_decode($game->metadata);

        if (isset($blocksToBeAdded)) {
            foreach($gameMetadata->users as $key => $metadataUser) {
                if ($metadataUser->currentPosition < 50) {
                    $userKeys[$i++] = $key;
                }

                if ($key == $user->id) {
                    ($metadataUser->currentPosition + $blocksToBeAdded) > 50 ? $metadataUser-
>currentPosition = 50 : $metadataUser->currentPosition += $blocksToBeAdded;
                    $metadataUser->coordinates = Game::$squares[$metadataUser-
>currentPosition]['position'];
                    $metadataUser->score += $blocksToBeAdded;
                }
            }
            if (isset($activeUser)) {
                $previousActiveUser = 0;
                foreach($gameMetadata->users as $key => $metadataUser) {
                    if ($metadataUser->activeUser == true) {
                        $metadataUser->activeUser = false;
                        $previousActiveUser = $key;
                    }
                }

                $indexNextActiveUser = array_search($previousActiveUser, $userKeys) + 1;
                if ($indexNextActiveUser >= sizeof($userKeys)) {
                    $gameMetadata->users->{$userKeys[0]}->activeUser = true;
                } else {
                    $gameMetadata->users->{$userKeys[$indexNextActiveUser]}->activeUser =
true;
                }
            }
            foreach($gameMetadata->users as $metadataUser) {
                if ($metadataUser->currentPosition == 50) {
                   $playersFinish += 1;
                }
            }

           if ($playersFinish == $gameMetadata->gameStats->numPlayers) {
               $gameMetadata->gameStats->gameFinished = true;
           }

        } elseif ($gameStarted) {
            $gameMetadata->gameStats->gameStarted = true;

        } elseif ($jumpTo) {
            if (isset($gameMetadata->users->{$user->id})) {
                $metadataUser = $gameMetadata->users->{$user->id};
```

```
                $isNewPositionSmaller = $metadataUser->currentPosition > $jumpTo;
                $metadataUser->currentPosition = $jumpTo;
                $metadataUser->coordinates = Game::$squares[$metadataUser-
>currentPosition]['position']; //check if is right
                if (is_null($pointFromActionCard)) {
                    $metadataUser->score = $isNewPositionSmaller ? $metadataUser->score - 2 :
$metadataUser->score + 2;
                } else {
                    $metadataUser->score += $pointFromActionCard;
                }
                $gameMetadata->users->{$user->id} = $metadataUser;
            }
        }

        $game->metadata = json_encode($gameMetadata);
        $game->lock = false;
        $toReturn = json_encode(['metadata' => $gameMetadata, 'status' => md5($game-
>metadata), "stopExtra" => Input::get('stopExtra')]);
        $game->save();

        return $toReturn;
    }

    function getQuestionAction($gameID, $category) {

        do {
            $game = Game::find($gameID);
            sleep(0.2);
        } while($game->lock);

        $game->lock = true;
        $game->save();
        $points = $category == 'hard' ? 2 : 1;
        $gameMetadata = json_decode($game->metadata);
        $question = Question::where('points', $points)->whereNotIn('id', $gameMetadata-
>gameStats->questionCardsPlayed)->orderByRaw("RAND()")->first();
        $gameMetadata->gameStats->questionCardsPlayed[] = $question->id;
        $game->metadata = json_encode($gameMetadata);
        $game->lock = false;
        $game->save();

        return $question->toJson();
    }

    function getActionCardAction() {
        $user = Session::get('user');
        do {
            $game = Game::find(Session::get('gameID'));
            sleep(0.2);
        } while($game->lock);

        $game->lock = true;
        $game->save();
        $gameMetadata = json_decode($game->metadata);
        if (isset($gameMetadata->users->{$user->id})) {
            $metadataUser = $gameMetadata->users->{$user->id};
                if (count($metadataUser->actionCards) == 9) {
                    $actionCard = GameCard::firstOrFail();
                    $metadataUser->actionCards[] = $actionCard->typeID;
                } else {
                    $actionCard = GameCard::whereNotIn('typeID', $metadataUser->actionCards)-
>orderByRaw("RAND()")->first();
                    $metadataUser->actionCards[] = $actionCard->typeID;
                }
        }
        $game->metadata = json_encode($gameMetadata);
```

```
        $game->lock = false;
        $game->save();
        return $actionCard;
    }

    function removeActionCardFromUserAction($typeID) {
        $user = Session::get('user');
        do {
            $game = Game::find(Session::get('gameID'));
            sleep(0.2);
        } while($game->lock);

        $game->lock = true;
        $game->save();
        $gameMetadata = json_decode($game->metadata);
        if (isset($gameMetadata->users->{$user->id})) {
            $metadataUser = $gameMetadata->users->{$user->id};
            foreach ($metadataUser->actionCards as $key => $actionCardTypeID) {
                if ($actionCardTypeID == $typeID) {
                    unset($metadataUser->actionCards[$key]);
                    break;
                }
            }
            $metadataUser->actionCards = array_values($metadataUser->actionCards);
        }
        $game->metadata = json_encode($gameMetadata);
        $game->lock = false;
        $game->save();
        return ['jhhjhg' => 'true'];
    }

    function validateAnswerAction($questionID, $actionType) {
        $givenAnswer = Input::get('answer');
        $question = Question::find($questionID);

        $result = $givenAnswer == $question->rightAnswer;
        if ($result) {
            $user = Session::get('user');
            do {
                $game = Game::find(Session::get('gameID'));
                sleep(0.2);
            } while($game->lock);

            $game->lock = true;
            $game->save();
            $gameMetadata = json_decode($game->metadata);
            if (isset($gameMetadata->users->{$user->id})) {
                $metadataUser = $gameMetadata->users->{$user->id};
                $metadataUser->score += ($actionType == 9 ? $question->points * 2 : $question-
>points);
                $gameMetadata->users->{$user->id} = $metadataUser;
            }
            $game->metadata = json_encode($gameMetadata);
            $game->lock = false;
            $game->save();
        }

        $result = ['givenAnswer' => $givenAnswer, 'rightAnswer' => $question->rightAnswer];

        return $result;
    }
}
```

IndexController.php

```
<?php

class IndexController extends BaseController {

    public $layout = 'layouts.layout';

      public function indexAction()
      {
        return View::make('index');
      }

      public function helpAction()
      {
        return View::make('help');
      }

}
```

2014_12_01_175806_create_table_users.php

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTableUsers extends Migration {

      /**
       * Run the migrations.
       *
       * @return void
       */
      public function up()
      {
        // Creates the Users table
         Schema::create('Users', function (Blueprint $table) {
             $table->increments('id');
             $table->string('nickname', 255);
             $table->integer('highScore')->unsigned();
             $table->timestamps();
             $table->engine = 'InnoDB';
         });
      }

      /**
       * Reverse the migrations.
       *
       * @return void
       */
      public function down()
      {
         Schema::dropIfExists('Users');
      }

}
```

2014_12_01_175822_create_table_game.php

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTableGame extends Migration {

    /**
       * Run the migrations.
       *
       * @return void
       */
      public function up()
      {
        // Creates the Game table
            Schema::create('Game', function (Blueprint $table) {
                $table->increments('id');
                $table->string('name', 255);
                $table->boolean('isPrivate')->default(false);
                $table->string('password', 255);
                $table->timestamps();
                $table->engine = 'InnoDB';
            });
      }

      /**
       * Reverse the migrations.
       *
       * @return void
       */
      public function down()
      {
            Schema::dropIfExists('Game');
      }

}
```

2014_12_01_175831_create_table_usergames.php

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTableUserGames extends Migration {

      /**
       * Run the migrations.
       *
       * @return void
       */
      public function up()
      {
        Schema::create('User_Game', function (Blueprint $table) {
            $table->integer('userID')->unsigned();
            $table->integer('gameID')->unsigned();
            $table->boolean('isHost')->default(false);
            $table->engine = 'InnoDB';
        });
      }

      /**
       * Reverse the migrations.
       *
       * @return void
       */
      public function down()
      {
        Schema::dropIfExists('User_Game');
      }

}
```

2014_12_05_180250_create_table_questions.php

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTableQuestions extends Migration {

      /**
       * Run the migrations.
       *
       * @return void
       */
      public function up()
      {
        Schema::create('Questions', function (Blueprint $table) {
            $table->increments('id');
            $table->text('question');
            $table->text('answer');
            $table->integer('points')->unsigned();
            $table->engine = 'InnoDB';
        });
      }

      /**
       * Reverse the migrations.
       *
       * @return void
       */
      public function down()
      {
        Schema::dropIfExists('Questions');
      }

}
```

2014_12_05_180821_create_table_gamecards.php

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTableGameCards extends Migration {

      /**
       * Run the migrations.
       *
       * @return void
       */
      public function up()
      {
        Schema::create('Game_Cards', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('typeID');
            $table->text('description');
            $table->engine = 'InnoDB';
        });
      }

      /**
       * Reverse the migrations.
       *
       * @return void
       */
      public function down()
      {
        Schema::dropIfExists('Game_Cards');
      }

}
```

2015_01_04_215834_add_column_metadata_to_Game_table.php

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class AddColumnMetadataToGameTable extends Migration {

    /**
       * Run the migrations.
       *
       * @return void
       */
      public function up()
      {
        Schema::table('Game', function (Blueprint $table) {
                $table->text('metadata');
        });
      }

      /**
       * Reverse the migrations.
       *
       * @return void
       */
      public function down()
      {
        Schema::table('Game', function($table)
        {
            $table->dropColumn('metadata');
        });
      }
}
```

2015_02_14_162316_add_proper_columns_and_remove_others_to_questions_table.php

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class AddProperColumnsAndRemoveOthersToQuestionsTable extends Migration {

      /**
       * Run the migrations.
       *
       * @return void
       */
      public function up()
      {
        Schema::table('Questions', function (Blueprint $table) {
            $table->text('answerOne')->nullable()->default(null);
            $table->text('answerTwo')->nullable()->default(null);
            $table->text('answerThree')->nullable()->default(null);
            $table->text('answerFour')->nullable()->default(null);
            $table->integer('rightAnswer');
            });
      }

      /**
       * Reverse the migrations.
       *
       * @return void
       */
      public function down()
      {
        Schema::table('Questions', function($table)
        {
                $table->dropColumn('answerOne');
                $table->dropColumn('answerTwo');
                $table->dropColumn('answerThree');
                $table->dropColumn('answerFour');
                $table->dropColumn('rightAnswer');
            });
      }
}
```

ActionCardSeeder.php

```
<?php

class ActionCardSeeder extends Seeder {

    public function run()
    {
        DB::table('Game_Cards')->delete();

        GameCard::create(['typeID' => 1, 'description' => 'With this card you can avoid following the snake after rolling the dice.']);
        GameCard::create(['typeID' => 2, 'description' => 'With this card you can avoid following the ladder after rolling the dice.']);
        GameCard::create(['typeID' => 3, 'description' => 'With this card you can move forward 3 blocks when is your turn and after rolling the dice, also 3 points will be added to your score.']);
        GameCard::create(['typeID' => 4, 'description' => 'With this card you can move forward 5 blocks when is your turn and after rolling the dice, also 5 points will be added to your score.']);
        GameCard::create(['typeID' => 5, 'description' => 'With this card you can move forward 3 blocks when is your turn and after rolling the dice, no points will be added to your score.']);
        GameCard::create(['typeID' => 6, 'description' => 'With this card you can move backward 3 blocks when is your turn and after rolling the dice, no points will be removed from your score.']);
        GameCard::create(['typeID' => 7, 'description' => 'With this card you can move backward 5 blocks when is your turn and after rolling the dice, no points will be removed from your score.']);
        GameCard::create(['typeID' => 8, 'description' => 'With this card you can move backward 3 blocks when is your turn and after rolling the dice, also 3 points will be removed to your score.']);
        GameCard::create(['typeID' => 9, 'description' => 'With this card you can double your points when answering a question correct.']);
    }
}
```

DatabaseSeeder.php

```
<?php

class DatabaseSeeder extends Seeder {

      /**
       * Run the database seeds.
       *
       * @return void
       */
      public function run()
      {
        Eloquent::unguard();

        $this->call('QuestionSeeder');
        $this->call('ActionCardSeeder');
      }

}
```

QuestionSeeder.php

```
<?php

class QuestionSeeder extends Seeder {

    public function run()
    {
        DB::table('Questions')->truncate();

        Question::create(['question' => 'How many Kilobytes is one Megabyte', 'points' => 1,
'answerOne' => '1000', 'answerTwo' => '100', 'answerThree' => '1024', 'answerFour' =>
'1048576', 'rightAnswer' => 3]);
        Question::create(['question' => 'The brain of any computer system is', 'points' => 1,
'answerOne' => 'ALU', 'answerTwo' => 'Memory', 'answerThree' => 'CPU', 'answerFour' =>
'Control Unit', 'rightAnswer' => 3]);
        Question::create(['question' => 'The binary system uses powers of', 'points' => 1,
'answerOne' => '2', 'answerTwo' => '10', 'answerThree' => '8', 'answerFour' => '16',
'rightAnswer' => 1]);
        Question::create(['question' => 'The time required for the fetching and execution of
one simple machine instruction is', 'points' => 1, 'answerOne' => 'Delay time', 'answerTwo' =>
'CPU cycle', 'answerThree' => 'Real time', 'answerFour' => 'Seek time', 'rightAnswer' => 2]);
        Question::create(['question' => 'ASCII stands for', 'points' => 1, 'answerOne' =>
'American standard code for information interchange', 'answerTwo' => 'All purpose scientific
code for information interchange', 'answerThree' => 'American security code for information
interchange', 'answerFour' => 'American Scientific code for information interchange',
'rightAnswer' => 1]);
        Question::create(['question' => 'Which device of computer operation dispenses with the
use of the keyboard?', 'points' => 1, 'answerOne' => 'Joystick', 'answerTwo' => 'Light pen',
'answerThree' => 'Mouse', 'answerFour' => 'Touch', 'rightAnswer' => 3]);
        Question::create(['question' => 'Any storage device added to a computer beyond the
immediately usable main storage is known as', 'points' => 1, 'answerOne' => 'Floppy disk',
'answerTwo' => 'Hard disk', 'answerThree' => 'Backing store', 'answerFour' => 'Punched card',
'rightAnswer' => 3]);
        Question::create(['question' => 'Which output device is used for translating
information from a computer into pictorial form on paper.', 'points' => 1, 'answerOne' =>
'Mouse', 'answerTwo' => 'Plotter', 'answerThree' => 'Touch panel', 'answerFour' => 'Card
punch', 'rightAnswer' => 2]);
        Question::create(['question' => 'The list of coded instructions is called', 'points'
=> 1, 'answerOne' => 'Computer program', 'answerTwo' => 'Algorithm', 'answerThree' =>
'Flowchart', 'answerFour' => 'Utility programs', 'rightAnswer' => 1]);
        Question::create(['question' => 'Any device that performs signal conversion is',
'points' => 1, 'answerOne' => 'Modulator', 'answerTwo' => 'Modem', 'answerThree' =>
'Keyboard', 'answerFour' => 'Plotter', 'rightAnswer' => 1]);
        Question::create(['question' => 'The complete picture of data stored in database is
known as', 'points' => 1, 'answerOne' => 'Record', 'answerTwo' => 'Schema', 'answerThree' =>
'System flowchart', 'answerFour' => 'DBMS', 'rightAnswer' => 2]);
        Question::create(['question' => 'A language which is close to that used within the
computer is', 'points' => 1, 'answerOne' => 'High-level language', 'answerTwo' => 'Assembly
language', 'answerThree' => 'Low-level language', 'answerFour' => 'All of the above',
'rightAnswer' => 3]);
        Question::create(['question' => 'A modern digital computer has', 'points' => 1,
'answerOne' => 'Extremely high speed', 'answerTwo' => 'Large memory', 'answerThree' => 'Almost
unlimited array', 'answerFour' => 'All of the above', 'rightAnswer' => 4]);
        Question::create(['question' => 'Compilers and interpreters are themselves', 'points'
=> 1, 'answerOne' => 'High-level language', 'answerTwo' => 'Codes', 'answerThree' =>
'Programs', 'answerFour' => 'Mnemonics', 'rightAnswer' => 3]);
        Question::create(['question' => 'A compiler means', 'points' => 1, 'answerOne' => 'A
person who compiles source programs', 'answerTwo' => 'The same thing as a programmer',
'answerThree' => 'Keypunch operator', 'answerFour' => 'A program which translates source
program into object program', 'rightAnswer' => 4]);
        Question::create(['question' => 'Computer memory consists of', 'points' => 1,
'answerOne' => 'RAM', 'answerTwo' => 'ROM', 'answerThree' => 'PROM', 'answerFour' => 'All of
```

```
the above', 'rightAnswer' => 4]);
        Question::create(['question' => 'The organization and interconnection of the various
components of a computer system is', 'points' => 1, 'answerOne' => 'Architecture', 'answerTwo'
=> 'Networks', 'answerThree' => 'Graphics', 'answerFour' => 'Designing', 'rightAnswer' => 1]);
        Question::create(['question' => 'As compared to diskettes, the hard disks are',
'points' => 1, 'answerOne' => 'more expensive', 'answerTwo' => 'more portable', 'answerThree'
=> 'less rigid', 'answerFour' => 'slowly accessed', 'rightAnswer' => 1]);
        Question::create(['question' => 'A billionth of a second is defined as a:', 'points'
=> 1, 'answerOne' => 'millisecond', 'answerTwo' => 'microsecond', 'answerThree' =>
'nanosecond', 'answerFour' => 'picoseconds', 'rightAnswer' => 3]);
        Question::create(['question' => 'Which of the following is not currently a topic in
computer science?', 'points' => 1, 'answerOne' => 'Speech recognition', 'answerTwo' =>
'Artificial intelligence', 'answerThree' => 'Thermodynamics', 'answerFour' =>
'Multiprocessing', 'rightAnswer' => 3]);
        Question::create(['question' => 'The unit of a computer system that executes program,
communicates with and often controls the operation of other subsystems of the computer is
known as', 'points' => 1, 'answerOne' => 'CPU', 'answerTwo' => 'Control Unit', 'answerThree'
=> 'I/O unit', 'answerFour' => 'Peripheral unit', 'rightAnswer' => 1]);
        Question::create(['question' => 'The brain of any computer system is', 'points' => 1,
'answerOne' => 'ALU', 'answerTwo' => 'Memory', 'answerThree' => 'CPU', 'answerFour' =>
'Control Unit', 'rightAnswer' => 3]);
        Question::create(['question' => 'Time during which a job is processed by the computer
is', 'points' => 1, 'answerOne' => 'Delay time', 'answerTwo' => 'Real time', 'answerThree' =>
'Execution time', 'answerFour' => 'Down time', 'rightAnswer' => 3]);
        Question::create(['question' => 'What is meant by a dedicated computer', 'points' =>
1, 'answerOne' => 'Which is used by one person only', 'answerTwo' => 'Which is assigned one
and only one task', 'answerThree' => 'Which uses on kind of software', 'answerFour' => 'Which
is meant for application software only', 'rightAnswer' => 2]);
        Question::create(['question' => 'Programs designed to perform specific tasks is also
known as', 'points' => 1, 'answerOne' => 'System software', 'answerTwo' => 'Application
software', 'answerThree' => 'Utility programs', 'answerFour' => 'Operating system',
'rightAnswer' => 2]);
        Question::create(['question' => 'A monitors _____ is the distance between the holes in
the mask behind the screen. This helps determine how sharp the dots appear', 'points' => 1,
'answerOne' => 'refresh rate', 'answerTwo' => 'dot pitch', 'answerThree' => 'resolution',
'answerFour' => 'color depth', 'rightAnswer' => 2]);
        Question::create(['question' => 'A multi programming system is one that can', 'points'
=> 1, 'answerOne' => 'run very fast', 'answerTwo' => 'share hardware resources with many
programs simultaneously', 'answerThree' => 'compute many programs simultaneously',
'answerFour' => 'use many operating systems', 'rightAnswer' => 2]);
        Question::create(['question' => '1024 bytes represent a', 'points' => 1, 'answerOne'
=> 'Megabyte', 'answerTwo' => 'Gigabyte', 'answerThree' => 'Kilobyte', 'answerFour' => 'None
of the above', 'rightAnswer' => 3]);
        Question::create(['question' => 'The function of CPU is', 'points' => 1, 'answerOne'
=> 'to provide a hard copy', 'answerTwo' => 'to read, interpret and process the information
and instruction', 'answerThree' => 'to communicate with the operator', 'answerFour' => 'to
provide external storage of text', 'rightAnswer' => 2]);
        Question::create(['question' => 'The central processor of a modern digital computer
consists of', 'points' => 1, 'answerOne' => 'control unit', 'answerTwo' => 'primary memory',
'answerThree' => 'control unit and primary memory', 'answerFour' => 'All of the above',
'rightAnswer' => 3]);
        Question::create(['question' => 'Dot-matrix is a type of', 'points' => 1, 'answerOne'
=> 'Tape', 'answerTwo' => 'Printer', 'answerThree' => 'Disk', 'answerFour' => 'Bus',
'rightAnswer' => 2]);
        Question::create(['question' => 'Where does a computer add and compare data?',
'points' => 1, 'answerOne' => 'Hard disk', 'answerTwo' => 'Floppy disk', 'answerThree' => 'CPU
chip', 'answerFour' => 'Memory chip', 'rightAnswer' => 3]);
        Question::create(['question' => 'General purpose computers are those that can be
adopted to countless uses simply by changing its', 'points' => 1, 'answerOne' => 'keyboard',
'answerTwo' => 'printer', 'answerThree' => 'program', 'answerFour' => 'display screen',
'rightAnswer' => 3]);
        Question::create(['question' => 'A computer program consists of', 'points' => 1,
'answerOne' => 'System flowchart', 'answerTwo' => 'Program flowchart', 'answerThree' =>
'Algorithms written in computers language', 'answerFour' => 'Discrete logical steps.',
'rightAnswer' => 3]);
        Question::create(['question' => 'Which of the following is not a part of the CPU',
```

```
'points' => 1, 'answerOne' => 'storage unit', 'answerTwo' => 'arithmetic and logic unit',
'answerThree' => 'program unit', 'answerFour' => 'control unit', 'rightAnswer' => 3]);
        Question::create(['question' => 'In this world of fast changing computer technology,
one of the most important factor to be considered while purchasing a PC is, it should have a
provision for', 'points' => 1, 'answerOne' => 'high level integration', 'answerTwo' => 'self
upgradability', 'answerThree' => 'intelligent sensors', 'answerFour' => 'faster data access',
'rightAnswer' => 2]);
        Question::create(['question' => 'A collection of eight bits is called', 'points' => 1,
'answerOne' => 'Byte', 'answerTwo' => 'Word', 'answerThree' => 'Record', 'answerFour' =>
'File', 'rightAnswer' => 1]);
        Question::create(['question' => 'Which is a secondary memory device?', 'points' => 1,
'answerOne' => 'CPU', 'answerTwo' => 'ALU', 'answerThree' => 'Floppy disk', 'answerFour' =>
'Mouse', 'rightAnswer' => 3]);
        Question::create(['question' => 'Can you tell what passes into and out from the
computer via its ports?', 'points' => 1, 'answerOne' => 'Data', 'answerTwo' => 'Bytes',
'answerThree' => 'Graphics', 'answerFour' => 'Pictures', 'rightAnswer' => 2]);
        Question::create(['question' => 'Which of the following input/output devices is not
associated with personal computers?', 'points' => 1, 'answerOne' => 'mice', 'answerTwo' =>
'color monitors', 'answerThree' => 'punched cards', 'answerFour' => 'dot-matrix printers',
'rightAnswer' => 3]);
        Question::create(['question' => 'The heart of any computer is the', 'points' => 1,
'answerOne' => 'CPU', 'answerTwo' => 'Memory', 'answerThree' => 'I/O Unit', 'answerFour' =>
'Disks', 'rightAnswer' => 1]);
        Question::create(['question' => 'The process of writing computer instructions in a
programming language is known as', 'points' => 1, 'answerOne' => 'Coding', 'answerTwo' =>
'Processing', 'answerThree' => 'Programming', 'answerFour' => 'File', 'rightAnswer' => 1]);
        Question::create(['question' => 'The most common input device used today is', 'points'
=> 1, 'answerOne' => 'Keyboard', 'answerTwo' => 'Track ball', 'answerThree' => 'Scanner',
'answerFour' => 'Mouse', 'rightAnswer' => 1]);
        Question::create(['question' => 'The desirable characteristic(s) of a memory system is
(are)', 'points' => 1, 'answerOne' => 'speed and reliability', 'answerTwo' => 'low power
consumption', 'answerThree' => 'durability and compactness', 'answerFour' => 'All of the
above', 'rightAnswer' => 4]);
        Question::create(['question' => 'Which is not a factor when categorizing a computer?',
'points' => 1, 'answerOne' => 'Speed of the output device', 'answerTwo' => 'Cost of the
system', 'answerThree' => 'Capacity of the hard disk', 'answerFour' => 'Where it was
purchased', 'rightAnswer' => 4]);
        Question::create(['question' => 'Which of the following can be output by a computer?',
'points' => 1, 'answerOne' => 'graphics', 'answerTwo' => 'voice', 'answerThree' => 'text',
'answerFour' => 'All of the above', 'rightAnswer' => 4]);
        Question::create(['question' => 'Which company is the biggest player in the
microprocessor industry?', 'points' => 1, 'answerOne' => 'Motorola', 'answerTwo' => 'IBM',
'answerThree' => 'Intel', 'answerFour' => 'AMD', 'rightAnswer' => 3]);
        Question::create(['question' => 'Which of the following is used as "Input device" for
the computer?', 'points' => 1, 'answerOne' => 'Printer', 'answerTwo' => 'VDU', 'answerThree'
=> 'TV', 'answerFour' => 'Light pen', 'rightAnswer' => 4]);
        Question::create(['question' => 'Primary storage is _____ as compared to secondary
storage.', 'points' => 1, 'answerOne' => 'Slow and inexpensive', 'answerTwo' => 'Fast and
inexpensive', 'answerThree' => 'Fast and expensive', 'answerFour' => 'Slow and expensive',
'rightAnswer' => 3]);
        Question::create(['question' => 'HTML stands for Hypertext Markup Language', 'points'
=> 1, 'answerOne' => 'true', 'answerTwo' => 'false', 'rightAnswer' => 1]);
        Question::create(['question' => 'CSS stand for Cascading Style Show', 'points' => 1,
'answerOne' => 'true', 'answerTwo' => 'false', 'rightAnswer' => 2]);
        Question::create(['question' => 'Computer can not do anything without a', 'points' =>
1, 'answerOne' => 'Chip', 'answerTwo' => 'Memory', 'answerThree' => 'Output device',
'answerFour' => 'Program', 'rightAnswer' => 4]);
        Question::create(['question' => 'A digital device that processes data is known as',
'points' => 1, 'answerOne' => 'Data processor', 'answerTwo' => 'Data entry', 'answerThree' =>
'DBMS', 'answerFour' => 'Database', 'rightAnswer' => 1]);
        Question::create(['question' => 'Which of the following is NOT a primary storage
device?', 'points' => 1, 'answerOne' => 'Magnetic tape', 'answerTwo' => 'Magnetic disk',
'answerThree' => 'Optical disk', 'answerFour' => 'All of the above', 'rightAnswer' => 4]);
        Question::create(['question' => 'Instructions and memory addresses are represented
by', 'points' => 1, 'answerOne' => 'character codes', 'answerTwo' => 'binary codes',
'answerThree' => 'binary word', 'answerFour' => 'parity bit', 'rightAnswer' => 2]);
```

```
        Question::create(['question' => 'Which of the following translate back from machine
code something resembling the source language', 'points' => 1, 'answerOne' => 'Interpreter',
'answerTwo' => 'Compiler', 'answerThree' => 'Assembler', 'answerFour' => 'Decompiler',
'rightAnswer' => 4]);
        Question::create(['question' => 'A six - digit card field used for postal ZIP codes is
defined as', 'points' => 1, 'answerOne' => 'A letter field', 'answerTwo' => 'An alphabetic
field', 'answerThree' => 'A numeric field', 'answerFour' => 'An alphanumeric field',
'rightAnswer' => 4]);
        Question::create(['question' => 'The benefit of using computers are that', 'points' =>
1, 'answerOne' => 'Computers are very fast and can store huge amounts of data', 'answerTwo' =>
'Computers produce accurate output even when the input is incorrect', 'answerThree' =>
'Computers are designed to the inflexible', 'answerFour' => 'All of the above', 'rightAnswer'
=> 1]);
        Question::create(['question' => 'A(n) ____ device is any device that provides
information which is sent to the CPU.', 'points' => 1, 'answerOne' => 'input', 'answerTwo' =>
'output', 'answerThree' => 'memory', 'answerFour' => 'storage', 'rightAnswer' => 1]);
        Question::create(['question' => 'The range of frequencies available for data
transmission is known as', 'points' => 1, 'answerOne' => 'Baud', 'answerTwo' => 'Bandwidth',
'answerThree' => 'Byte', 'answerFour' => 'Bits', 'rightAnswer' => 2]);
        Question::create(['question' => 'The two main components of the CPU is', 'points' =>
1, 'answerOne' => 'Control unit and registers', 'answerTwo' => 'Control unit and ALU',
'answerThree' => 'Registers and main memory', 'answerFour' => 'ALU and bus', 'rightAnswer' =>
2]);
        Question::create(['question' => 'Which of the following does not affect the resolution
of a video display image?', 'points' => 1, 'answerOne' => 'bandwidth', 'answerTwo' => 'raster
scan rate', 'answerThree' => 'vertical and horizontal lines of resolution', 'answerFour' =>
'screen size', 'rightAnswer' => 4]);
        Question::create(['question' => 'A program used to detect overall system malfunction
is', 'points' => 1, 'answerOne' => 'System analysis', 'answerTwo' => 'System software',
'answerThree' => 'Utilities', 'answerFour' => 'System diagnostics', 'rightAnswer' => 4]);
        Question::create(['question' => 'A computer, by definition, is any device that
computers. This broad definition includes which of the following?', 'points' => 1, 'answerOne'
=> 'Calculators', 'answerTwo' => 'Cash registers', 'answerThree' => 'Desktop computers',
'answerFour' => 'All of the above', 'rightAnswer' => 4]);
        Question::create(['question' => 'The brain of any computer system is', 'points' => 1,
'answerOne' => 'ALU', 'answerTwo' => 'Memory', 'answerThree' => 'CPU', 'answerFour' =>
'Control Unit', 'rightAnswer' => 3]);
        Question::create(['question' => 'Which of the following languages is more suited to a
structured program?', 'points' => 2, 'answerOne' => 'FORTRAN', 'answerTwo' => 'BASIC',
'answerThree' => 'PASCAL', 'answerFour' => 'None of the above', 'rightAnswer' => 3]);
        Question::create(['question' => 'Which of the following computer language is used for
artificial intelligence?', 'points' => 2, 'answerOne' => 'FORTRAN', 'answerTwo' => 'PROLOG',
'answerThree' => 'C', 'answerFour' => 'COBOL', 'rightAnswer' => 2]);
        Question::create(['question' => 'Binary numbers need more places for counting
because', 'points' => 2, 'answerOne' => 'They are always big numbers', 'answerTwo' => 'Any no.
of 0s can be added in front of them', 'answerThree' => 'Binary base is small', 'answerFour' =>
'None of the above', 'rightAnswer' => 3]);
        Question::create(['question' => 'A single packet on a data link is known as', 'points'
=> 2, 'answerOne' => 'Path', 'answerTwo' => 'Frame', 'answerThree' => 'Block', 'answerFour' =>
'Group', 'rightAnswer' => 2]);
        Question::create(['question' => 'Which method is used to connect a remote computer?',
'points' => 2, 'answerOne' => 'Device', 'answerTwo' => 'Dialup', 'answerThree' =>
'Diagnostic', 'answerFour' => 'Logic circuit', 'rightAnswer' => 2]);
        Question::create(['question' => 'The symbols used in an assembly language are',
'points' => 2, 'answerOne' => 'Codes', 'answerTwo' => 'Mnemonics', 'answerThree' =>
'Assembler', 'answerFour' => 'All of the above', 'rightAnswer' => 2]);
        Question::create(['question' => 'Which language was devised by Dr. Seymour Aubrey
Papert?', 'points' => 2, 'answerOne' => 'APL', 'answerTwo' => 'COBOL', 'answerThree' =>
'LOGO', 'answerFour' => 'FORTRAN', 'rightAnswer' => 3]);
        Question::create(['question' => 'A group of magnetic tapes, videos or terminals
usually under the control of one master is', 'points' => 2, 'answerOne' => 'Cylinder',
'answerTwo' => 'Cluster', 'answerThree' => 'Surface', 'answerFour' => 'Track', 'rightAnswer'
=> 2]);
        Question::create(['question' => 'A type of channel used to connect a central processor
and peripherals which uses multipling is known as', 'points' => 2, 'answerOne' => 'Modem',
'answerTwo' => 'Network', 'answerThree' => 'Multiplexer', 'answerFour' => 'All of the above',
```

```
'rightAnswer' => 3]);
        Question::create(['question' => 'he device that can both feed data into and accept
data from a computer is', 'points' => 2, 'answerOne' => 'ALU', 'answerTwo' => 'CPU',
'answerThree' => 'Input-Output device', 'answerFour' => 'All of the above', 'rightAnswer' =>
3]);
        Question::create(['question' => 'Which of the following can store information in the
form of microscopic pits on metal disks.', 'points' => 2, 'answerOne' => 'Laser disks',
'answerTwo' => 'Tape cassettes', 'answerThree' => 'RAM cartridge', 'answerFour' => 'Punched
cards', 'rightAnswer' => 1]);
        Question::create(['question' => 'A notation used to express clearly on algorithm is
known as', 'points' => 2, 'answerOne' => 'Algorithmic language', 'answerTwo' => 'Assembly
language', 'answerThree' => 'Machine language', 'answerFour' => 'High level language',
'rightAnswer' => 1]);
        Question::create(['question' => 'Compression of digital data for efficient storage
is', 'points' => 2, 'answerOne' => 'Buffer', 'answerTwo' => 'CPU', 'answerThree' => 'Packing',
'answerFour' => 'Field', 'rightAnswer' => 3]);
        Question::create(['question' => 'A memory that does not change its contents without
external causes is known as', 'points' => 2, 'answerOne' => 'Dynamic memory', 'answerTwo' =>
'Static memory', 'answerThree' => 'RAM', 'answerFour' => 'EEPROM', 'rightAnswer' => 2]);
        Question::create(['question' => 'Which of the following is the coding of data so that
is cant be easily understood if intercepted.', 'points' => 2, 'answerOne' => 'Barcode',
'answerTwo' => 'Decoder', 'answerThree' => 'Encryption', 'answerFour' => 'Mnemonics',
'rightAnswer' => 3]);
        Question::create(['question' => 'RAM is used as a short memory because it is',
'points' => 2, 'answerOne' => 'Volatile', 'answerTwo' => 'Has small capacity', 'answerThree'
=> 'Is very expensive', 'answerFour' => 'Is programmable', 'rightAnswer' => 1]);
        Question::create(['question' => 'An index register that is automatically incremented
or decremented with each use is', 'points' => 2, 'answerOne' => 'Auto index', 'answerTwo' =>
'Asynchronous', 'answerThree' => 'Assembler', 'answerFour' => 'Compiler', 'rightAnswer' =>
1]);
        Question::create(['question' => 'Who is considered the "father" of the minicomputer
and one of the founder fathers of the modern computer industry world-wide?', 'points' => 2,
'answerOne' => 'George Tate', 'answerTwo' => 'Kenneth H. Olsen', 'answerThree' => 'Seymour
Cray', 'answerFour' => 'Basic Pascal', 'rightAnswer' => 2]);
        Question::create(['question' => 'An instruction that transfers program control to one
or more possible paths is known as', 'points' => 2, 'answerOne' => 'Utility program',
'answerTwo' => 'System software', 'answerThree' => 'Broadband channel', 'answerFour' =>
'Application program', 'rightAnswer' => 3]);
        Question::create(['question' => 'The ALU of a computer normally contains a number of
high speed storage elements called', 'points' => 2, 'answerOne' => 'semiconductor memory',
'answerTwo' => 'registers', 'answerThree' => 'hard disk', 'answerFour' => 'magnetic disk',
'rightAnswer' => 2]);
        Question::create(['question' => 'Which of the following holds data and processing
instructions temporarily until the CPU needs it?', 'points' => 2, 'answerOne' => 'ROM',
'answerTwo' => 'control unit', 'answerThree' => 'main memory', 'answerFour' => 'coprocessor
chips', 'rightAnswer' => 3]);
        Question::create(['question' => 'A high speed device used in CPU for temporary storage
during processing is called', 'points' => 2, 'answerOne' => 'register', 'answerTwo' => 'bus',
'answerThree' => 'databus', 'answerFour' => 'None of the above', 'rightAnswer' => 1]);
        Question::create(['question' => 'The first electronic digital computer contained?',
'points' => 2, 'answerOne' => 'Electronic valves', 'answerTwo' => 'Vacuum tubes',
'answerThree' => 'Transistors', 'answerFour' => 'Semiconductor memory', 'rightAnswer' => 1]);
        Question::create(['question' => 'The memory which is ultraviolet light erasable and
electrically programmable is', 'points' => 2, 'answerOne' => 'ROM', 'answerTwo' => 'PROM',
'answerThree' => 'RAM', 'answerFour' => 'EPROM', 'rightAnswer' => 4]);
    }

}
```

Game.php

```
<?php

class Game extends Eloquent {

      /**
       * The database table used by the model.
       *
       * @var string
       */
      protected $table = 'Game';
    protected $fillable = ['name', 'isPrivate', 'password', 'metadata'];
    protected $userNum = 0;

    public static $squares = [
        0 => ['position' => ['x' => 80, 'y'=> 625], 'jump' => false, 'question' => false,
'actionCard' => false],
        1 => ['position' => ['x' => 165, 'y'=> 625], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        2 => ['position' => ['x' => 242, 'y'=> 625], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        3 => ['position' => ['x' => 318, 'y'=> 625], 'jump' => false, 'question' => false,
'actionCard' => true],
        4 => ['position' => ['x' => 394, 'y'=> 625], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        5 => ['position' => ['x' => 471, 'y'=> 625], 'jump' => false, 'question' => 'hard',
'actionCard' => false],
        6 => ['position' => ['x' => 547, 'y'=> 625], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        7 => ['position' => ['x' => 623, 'y'=> 625], 'jump' => false, 'question' => false,
'actionCard' => true],
        8 => ['position' => ['x' => 700, 'y'=> 625], 'jump' => false, 'question' => 'hard',
'actionCard' => false],
        9 => ['position' => ['x' => 780, 'y'=> 625], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        10 => ['position' => ['x' => 780, 'y'=> 552], 'jump' => false, 'question' => false,
'actionCard' => true],
        11 => ['position' => ['x' => 780, 'y'=> 479], 'jump' => false, 'question' => false,
'actionCard' => false],
        12 => ['position' => ['x' => 780, 'y'=> 405], 'jump' => false, 'question' => false,
'actionCard' => true],
        13 => ['position' => ['x' => 780, 'y'=> 331], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        14 => ['position' => ['x' => 780, 'y'=> 258], 'jump' => 11, 'question' => false,
'actionCard' => false],
        15 => ['position' => ['x' => 780, 'y'=> 184], 'jump' => 20, 'question' => false,
'actionCard' => false],
        16 => ['position' => ['x' => 780, 'y'=> 111], 'jump' => false, 'question' => false,
'actionCard' => true],
        17 => ['position' => ['x' => 780, 'y'=> 40], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        18 => ['position' => ['x' => 700, 'y'=> 40], 'jump' => false, 'question' => false,
'actionCard' => true],
        19 => ['position' => ['x' => 624, 'y'=> 40], 'jump' => false, 'question' => 'hard',
'actionCard' => false],
        20 => ['position' => ['x' => 548, 'y'=> 40], 'jump' => false, 'question' => false,
'actionCard' => false],
        21 => ['position' => ['x' => 472, 'y'=> 40], 'jump' => false, 'question' => false,
'actionCard' => true],
        22 => ['position' => ['x' => 395, 'y'=> 40], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        23 => ['position' => ['x' => 319, 'y'=> 40], 'jump' => 28, 'question' => false,
'actionCard' => false],
```

```
        24 => ['position' => ['x' => 243, 'y'=> 40], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        25 => ['position' => ['x' => 167, 'y'=> 40], 'jump' => false, 'question' => false,
'actionCard' => true],
        26 => ['position' => ['x' => 167, 'y'=> 113], 'jump' => false, 'question' => false,
'actionCard' => false],
        27 => ['position' => ['x' => 167, 'y'=> 187], 'jump' => false, 'question' => 'hard',
'actionCard' => false],
        28 => ['position' => ['x' => 167, 'y'=> 260], 'jump' => false, 'question' => false,
'actionCard' => false],
        29 => ['position' => ['x' => 167, 'y'=> 333], 'jump' => false, 'question' => false,
'actionCard' => true],
        30 => ['position' => ['x' => 167, 'y'=> 406], 'jump' => 26, 'question' => false,
'actionCard' => false],
        31 => ['position' => ['x' => 167, 'y'=> 479], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        32 => ['position' => ['x' => 243, 'y'=> 479], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        33 => ['position' => ['x' => 319, 'y'=> 479], 'jump' => false, 'question' => false,
'actionCard' => true],
        34 => ['position' => ['x' => 395, 'y'=> 479], 'jump' => false, 'question' => 'hard',
'actionCard' => false],
        35 => ['position' => ['x' => 472, 'y'=> 479], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        36 => ['position' => ['x' => 548, 'y'=> 479], 'jump' => 40, 'question' => false,
'actionCard' => false],
        37 => ['position' => ['x' => 624, 'y'=> 479], 'jump' => false, 'question' => false,
'actionCard' => true],
        38 => ['position' => ['x' => 624, 'y'=> 406], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        39 => ['position' => ['x' => 624, 'y'=> 333], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        40 => ['position' => ['x' => 624, 'y'=> 260], 'jump' => false, 'question' => false,
'actionCard' => false],
        41 => ['position' => ['x' => 624, 'y'=> 187], 'jump' => false, 'question' => false,
'actionCard' => false],
        42 => ['position' => ['x' => 548, 'y'=> 187], 'jump' => false, 'question' => false,
'actionCard' => true],
        43 => ['position' => ['x' => 472, 'y'=> 187], 'jump' => 47, 'question' => false,
'actionCard' => false],
        44 => ['position' => ['x' => 395, 'y'=> 187], 'jump' => false, 'question' => 'hard',
'actionCard' => false],
        45 => ['position' => ['x' => 319, 'y'=> 187], 'jump' => 41, 'question' => false,
'actionCard' => false],
        46 => ['position' => ['x' => 319, 'y'=> 260], 'jump' => false, 'question' => false,
'actionCard' => false],
        47 => ['position' => ['x' => 319, 'y'=> 333], 'jump' => false, 'question' => false,
'actionCard' => false],
        48 => ['position' => ['x' => 395, 'y'=> 333], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        49 => ['position' => ['x' => 472, 'y'=> 333], 'jump' => 46, 'question' => false,
'actionCard' => false],
        50 => ['position' => ['x' => 472, 'y'=> 260], 'jump' => false, 'question' => 'easy',
'actionCard' => false],
        51 => ['position' => ['x' => 472, 'y'=> 113], 'jump' => false, 'question' => false,
'actionCard' => false]
    ];

    function users()
      {
        return $this->belongsToMany('User', 'User_Game', 'gameID', 'userID');
      }

    function addGameToSession()
    {
        Session::put("gameID", $this->id);
```

```
    }

    function countUsers()
    {
        return $this->users->count();
    }

}
```

GameCard.php

```
<?php

class GameCard extends Eloquent {

      /**
       * The database table used by the model.
       *
       * @var string
       */
    public $timestamps = false;
      protected $table = 'Game_Cards';

    public static $actionCards = [
        1 => ['description' => 'With this card you can avoid following the snake.'],
        2 => ['description' => 'With this card you can avoid following the ladder.'],
        3 => ['description' => 'With this card you can move forward 3 blocks when is your turn
and after rolling the dice, also 3 points will be added to your score.'],
        4 => ['description' => 'With this card you can move forward 5 blocks when is your turn
and after rolling the dice, also 5 points will be added to your score.'],
        5 => ['description' => 'With this card you can move forward 3 blocks when is your turn
and after rolling the dice, no points will be added to your score.'],
        6 => ['description' => 'With this card you can move backward 3 blocks when is your
turn and after rolling the dice, no points will be removed from your score.'],
        7 => ['description' => 'With this card you can move backward 5 blocks when is your
turn and after rolling the dice, no points will be removed from your score.'],
        8 => ['description' => 'With this card you can move backward 3 blocks when is your
turn and after rolling the dice, also 3 points will be removed to your score.'],
        9 => ['description' => 'With this card you can double your points when answering a
question correct.'],
    ];
}
```

Question.php

```
<?php

class Question extends Eloquent {

      /**
       * The database table used by the model.
       *
       * @var string
       */
    public $timestamps = false;
      protected $table = 'Questions';



}
```

User.php

```
<?php

class User extends Eloquent {

      /**
       * The database table used by the model.
       *
       * @var string
       */
      protected $table = 'Users';
    protected $fillable = ['nickname'];

      function users()
      {
        return $this->belongsToMany('Games', 'User_Game', 'userID', 'gameID');
      }

    function addUserToSession() {
        Session::put("user", $this);
    }
}
```

GameControllerTest.php

```
<?php

class GameControllerTest extends TestCase {

      /**
       * A basic functional test example.
       *
       * @return void
       */
      public function testBasicExample()
      {
        $crawler = $this->client->request('GET', '/');

        $this->assertTrue($this->client->getResponse()->isOk());
      }

    function testSearchActionReturnProperJson()
    {
        $user = new User(['nickname' => 'Bill', 'highScore' => 0]);
        $user->save();
        $game = new Game(['name' => 'Test', 'password' => '']);
        $game->save();
        $game->users()->attach($user->id);

        $response = $this->route('GET', 'search');
        $this->assertJson($response->getContent());
    }

    function testSearchActionReturnProperNumOfUserInEachGame()
    {
        $users[] = new User(['nickname' => 'Bill', 'highScore' => 0]);
        $users[] = new User(['nickname' => 'John', 'highScore' => 0]);
        $users[] = new User(['nickname' => 'Tsak', 'highScore' => 0]);
        $users[0]->save();
        $users[1]->save();
        $users[2]->save();

        $ids = [$users[0]->id, $users[1]->id, $users[2]->id];
        $game = new Game(['name' => 'Test', 'password' => '']);
        $game->save();
        $game->users()->sync($ids);

        $json = $this->route('GET', 'search')->getContent();
        $this->assertJson($json);

        $this->assertEquals(3, json_decode($json)[0]->usersNum);

    }

    function testCreateActionCreatesProperModels()
    {
        $requestData  = ['player-name' => 'Bill', 'game-name' => 'Foo'];

        $this->route('POST', 'create', [], $requestData)->getContent();
        $games = Game::all();

        $this->assertEquals(1, $games->count());
        $this->assertEquals(1, $games[0]->users->count());
        $this->assertEquals('Foo', $games[0]->name);
        $this->assertEquals('Bill', $games[0]->users[0]->nickname);
    }
```

```
    public function testCreateActionExists()
    {
        Route::enableFilters();

        $this->client->restart();
        $this->client->request('POST', 'game/create');
        $this->assertResponseStatus(200);
    }

}
```

GameTest.php

```
<?php

use Mockery as m;

class GameTest extends TestCase
{

      /**
       * @var Document
       */
      private $_document;

      /**
       * @var Student
       */
      private $_student;

//    function setUp()
//    {
//        $this->markTestIncomplete('sadsad');
//      parent::setUp();
//      $this->_document = Game::find(2);
//      $this->_student = $this->_student();
//    }

    function testDemoModel()
    {
        $game = new Game();
        $game->name = 'test1';
        $game->isPrivate = false;
        $game->password = 'sdfsf';
        $game->save();

        $actual = Game::find($game->id);

        $this->assertEquals($actual->name, $game->name);
    }
}
```

game.blade.php

```
/**
 * Created by PhpStorm.
 * User: vasgen
 * Date: 28/10/2014
 * Time: 4:45 μμ
 */<!doctype html>
 <html lang="en">
 <head>
      <meta charset="UTF-8">
      <title>Laravel PHP Framework</title>
     <link rel="stylesheet" href="assets/bootstrap.min.css">
     <style>
       body {padding-top:50px;padding-bottom:20px;}
     </style>
     <link rel="stylesheet" href="assets/bootstrap-theme.min.css">
     <link rel="stylesheet" href="assets/main.css">
     <link rel="stylesheet" href="assets/animate.min.css">
     <link rel="stylesheet" href="assets/hover-min.css" media="all">
     <link href='http://fonts.googleapis.com/css?family=Jura:400,600,500,300' rel='stylesheet'
type='text/css'>
     <script src="/vendors/modernizr-2.6.2-respond-1.1.0.min.js"></script>
     <script src="/vendors/jquery-1.11.1.min.js"></script>
     <script src="/vendors/jquery.lightbox_me.js"></script>
     <script src="/vendors/jquery.zoomooz.min.js"></script>
     <script src="/vendors/underscore-min.js"></script>
 </head>
 <body class="gamePanel">

     @if ($message = Session::get('message'))
      <div class="alert alert-warning" data-uk-alert>
             <a href="#" class="close" data-dismiss="alert">&times;</a>
       <p>
               @if(is_array($message))
                       @foreach ($message as $m)
                               {{ $m }}
                       @endforeach
               @else
                       {{ $message }}
               @endif
       </p>
      </div>
     @endif

     @if (count($errors->all()) > 0)
      <div class="uk-alert uk-alert-danger" data-uk-alert>
       <a class="uk-alert-close uk-close"></a>
       @if(is_array($errors->all()))
               @foreach ($errors->all() as $m)
                       <p>
                               {{ $m }}
                       </p>
               @endforeach
       @else
               {{ $errors }}
       @endif
      </div>
     @endif

     @yield('body')
 </body>
 </html>
```

layout.blade.php

```
<!doctype html>
<html lang="en">
<head>
      <meta charset="UTF-8">
      <title>GS</title>
    <link rel="stylesheet" href="/assets/bootstrap.min.css">
    <style>
      body {padding-top:50px;padding-bottom:20px;}
    </style>
    <link rel="stylesheet" href="/assets/bootstrap-theme.min.css">
    <link rel="stylesheet" href="/assets/main.css">
    <link rel="stylesheet" href="/assets/animate.min.css">
    <link rel="stylesheet" href="/assets/hover-min.css" media="all">
    <link href='http://fonts.googleapis.com/css?family=Jura:400,600,500,300' rel='stylesheet'
type='text/css'>
    <script src="/vendors/modernizr-2.6.2-respond-1.1.0.min.js"></script>
    <script src="/vendors/jquery-1.11.1.min.js"></script>
    <script src="/vendors/jquery.lightbox_me.js"></script>
    <script src="/vendors/jquery.zoomooz.min.js"></script>
    <script src="/vendors/modal.js"></script>
    <script src="/vendors/underscore-min.js"></script>
</head>
<body class="gamePanel">

    @if ($message = Session::get('message'))
      <div class="alert alert-warning" data-uk-alert>
            <a href="#" class="close" data-dismiss="alert">&times;</a>
        <p>
                @if(is_array($message))
                        @foreach ($message as $m)
                                {{ $m }}
                        @endforeach
                @else
                        {{ $message }}
                @endif
        </p>
      </div>
    @endif

    @if (count($errors->all()) > 0)
         <div class="alert alert-warning">
             <a href="#" class="close" data-dismiss="alert">&times;</a>
        @if(is_array($errors->all()))
                @foreach ($errors->all() as $m)
                        <p>
                          <strong>
                                  {{ $m }}
                          </strong>
                        </p>
                @endforeach
        @else
                {{ $errors }}
        @endif
         </div>
    @endif

    @yield('body')
</body>
</html>
```

action-card.blade.php

```
<script type="text/template" id="action-card-template">
    <div class="modal-dialog">
        <div class="modal-content whiteBlueBG">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span
                    aria-hidden="true">&times;</span></button>
                <strong><h4 class="modal-title" id="myModalLabel">ACTION CARD</h4></strong>
            </div>
            <div class="modal-body">
                <strong><p><%= actionData.description %></p></strong>
            </div>
            <div class="modal-footer">
                <button id="modal-action-btn" type="button" class="pull-left font20 btn-lg
btn-primary" data-dismiss="modal">OK</button>
            </div>
        </div>
    </div>
</script>
```

action-presentation.blade.php

```
<script type="text/template" id="use-action-cards-template">
    <div class="modal-dialog">
        <div class="modal-content whiteBlueBG">
            <div class="modal-header">
                <strong><h4 class="modal-title font25" id="myModalLabel">ACTION
CARDS</h4></strong>
            </div>
            <div class="modal-body">
                <table class="table table-hover table-striped table-responsive">
                <strong><h3>Select an action to use:</h3></strong>
                    <% _.each(actionCards, function(actionCard, key) { %>
                        <% if (_.indexOf(userActionCards, key) != -1) { %>
                            <input id="action-<%= key %>" type="radio" name="action" value=<%=
key %>>
                            <label id="label-<%= key %>" for="action-<%= key %>"><strong><%=
actionCard.description %> </strong></label>
                        <% } %>
                    <% }) %>
                <div class="modal-footer">
                    <button id="apply-action" type="button" class="pull-left font20 btn-lg
btn-primary">OK</button>
                </div>
            </div>
        </div>
    </div>
</script>
```

games.blade.php

```
<script type="text/template" id="game-results-template">
    <div class="table-responsive">
        <table class="table table-hover table-striped table-responsive">
            <thead>
                <tr>
                    <th>#</th>
                    <th>Name</th>
                    <th>Players</th>
                    <th>Private</th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                <% _.each(games, function(value, key) { %>
                <tr>
                    <th class="row"><%= key + 1 %></th>
                    <td><%= value.name %></td>
                    <td><%= value.usersNum %> / 4</td>
                    <td><%= (value.isPrivate == 1) ? "Yes" : "No" %></td>
                    <td>
                        <% if (value.usersNum < 4 && !value.gameStarted) { %>
                            <a class="join-game fontW800 green" disabled="disabled" data-
private="<%= value.isPrivate %>" data-id="<%= value.id %>" href="{{{ URL::route('join', ['id'
=> '']) }}}/<%= value.id %>">Join Game</a>
                        <% } %>
                    </td>
                </tr>
                <% }) %>
            </tbody>
        </table>
    </div>
</script>
```

question.blade.php

```
<script type="text/template" id="questions-template">
    <div class="modal-dialog">
        <div class="modal-content whiteBlueBG">
            <div class="modal-header">
                <strong><h4 class="modal-title font25"
id="myModalLabel">QUESTION</h4></strong>
            </div>
            <div class="modal-body">
                <strong><p><%= questionData.question %></p><stromg>
                <form id="questionForm" action="#">
                    <strong>
                    <input type="hidden" id="questionID" value="<%= questionData.id %>">
                    <input id="answer-1" type="radio" name="answer" value="1">
                    <label id="label-1" for="answer-1"><%= '  ' + questionData.answerOne
%></label>
                        <br>
                    <input id="answer-2" type="radio" name="answer" value="2">
                    <label id="label-2" for="answer-2"><%= '  ' + questionData.answerTwo
%></label>
                        <br>
                    <% if (questionData.answerThree) { %>
                        <input id="answer-3" type="radio" name="answer" value="3">
                        <label id="label-3" for="answer-3"><%= '  ' + questionData.answerThree
%></label>
                        <br>
                    <% } %>
                    <% if (questionData.answerFour) { %>
                        <input id="answer-4" type="radio" name="answer" value="4">
                        <label id="label-4" for="answer-4"><%= '  ' + questionData.answerFour
%></label>
                     <% } %>
                     </strong>
                     <div class="modal-footer">
                        <input type="submit" value="OK" id="btnQuestionAnswer" onsubmit=""
class="pull-left font20 btn-lg btn-primary">
                        <div id='answer-result'></div>
                     </div>
                </form>
            </div>
        </div>
    </div>
</script>
```

use-action-cards.blade.php

```
<script type="text/template" id="ask-for-action-template">
  <div class="modal-dialog">
    <div class="modal-content whiteBlueBG">
      <div class="modal-body">
        <strong><p class="font20">Would you like to use an Action Card?</p></strong>
      </div>
      <div class="modal-footer">
        <button id="use-action" type="button" class="pull-left btn-lg btn-primary font15 btn-
primary">Yes</button>
        <button id="no-use-action" type="button" class="pull-left btn-lg font15 btn-default"
data-dismiss="modal">No</button>
      </div>
    </div>
  </div>
</script>
```

user-info.blade.php

```
<script type="text/template" id="user-info-template">
    <% var count = 0 %>
    <% _.each(gameData.users, function(user, key) { %>
    <div class="row">
        <% count++ %>
        <div class="col-xs-11 col-xs-offset-1">
            <div id="player-info" class="row mb-5 pb-1 borderB <%= user.activeUser ?
"opacityPlayerNot" : "opacityPlayer" %>">
                <div class="row">
                    <div class="col-xs-12 col-sm-4 col-md-4">
                        <p class="text-center font35 bolder greenColor">
                        <strong><%=  user.name %></strong>
                        </p>
                        <div class="text-center">
                            <img src="/images/avatars/avatarBG0<%= count %>.png" class=""
alt="Player Avatar">
                        </div>
                    </div>
                    <div class="col-xs-12 col-sm-8 col-md-8 noPaddingLR">
                        <p class="font30 greenLight text-left hidden-sm hidden-md hidden-lg
mt-20- pl15">
                        </p>
                    </div>
                </div>
                <div class="col-xs-12 font35 bolder mt-50- noPaddingLR">
                    Score
                    <span class="badge font35 m-paddingLR greenBG">
                        <%=  user.score %>
                    </span>
                </div>
            </div>
        </div>
    </div>
    <% }) %>
</script>
```

wait-finish-game.blade.php

```
<script type="text/template" id="wait-finish-game-template">
    <div class="modal-dialog">
        <div class="modal-content whiteBlueBG">
            <div class="modal-body">
                <strong><p class="font15"><%=' ' + name %> you finish the game with <%=' ' +
score + ' '%> points, wait till all players finish to have the winner!!!</p><strong>
            </div>
        </div>
    </div>
</script>
```

win-game.blade.php

```
<script type="text/template" id="win-game-template">
    <div class="modal-dialog">
        <div class="modal-content whiteBlueBG">
            <div class="modal-body">
                <strong><p class="font25">User <%=' ' + name %>, won the game with <%=' ' +
score + ' '%> points !!!</p><strong>
            </div>
            <div class="modal-footer">
                <button id="modal-action-btn" type="button" class="pull-left font20 btn-lg
btn-primary" data-dismiss="modal">OK</button>
            </div>
        </div>
    </div>
</script>
```

game.blade.php

```
@extends('layouts.layout')

@section('body')
@include('templates.user-info')
@include('templates.question')
@include('templates.action-card')
@include('templates.win-game')
@include('templates.action-presentation')
@include('templates.use-action-cards')
@include('templates.wait-finish-game')

<div class="modal fade" id="myModal" tabindex="-1" data-dismiss="modal" role="dialog" data-
backdrop="static" data-keyboard="false"></div>

<div id="wait-background"></div>

<div id="bar-1" class="timer-bar-main-container azure">
  <div class="timer-wrap">
    <div class="timer-bar-percentage" data-percentage="75"></div>
    <div class="timer-bar-container">
      <div class="timer-bar"></div>
    </div>
      <p id="login-players" class="font25">Wait all players to login</p>
  </div>
</div>

<div class="container-fluid">
    <div class="row">
        <div id="user-info-results" class="col-xs-2">
        </div>
        <div class="col-xs-8 text-center">
            <canvas id="canvas" class="img-responsive mAutoLR" width="946" height="659"
style="max-width:80%;">
            </canvas>
            <form>
                <input type="button" id="btnThrowDice" class="btn font60 bolder whiteColor
hvr-buzz-out dice-6 noneBorder" style="padding-bottom: 0px; padding-top: 0px; height: 155px;">
            </form>
        </div>
        <div class="col-xs-2">
            <div class="row">
                <div class="col-xs-12 cardBG noPaddingLR">
                    <div id="cardOneContainer">
                        <div id="cardOne" class="shadow">
                            <div class="front faceCardOne">
                                <img src="/images/card01.png" class="img-responsive">
                            </div>
                            <div class="back faceCardOne center">
                                <p>These Cards are for the Blocks that contains Questions and
will be used during the Game</p>
                            </div>
                        </div>
                    </div>
                    <p class="font30 greenLight mt-35 ml-7 displayNone">CARDS</p>
                </div>
                <div class="col-xs-12 cardBG noPaddingLR">
                    <div id="cardTwoContainer">
                        <div id="cardTwo" class="shadow">
                            <div class="front faceCardTwo">
                                <img src="/images/card02.png" class="img-responsive">
                            </div>
                            <div class="back faceCardTwo center">
```

```
                                <p>These Cards are for the Blocks that contains Actions that
can be used during the Game from each Player</p>
                            </div>
                        </div>
                    </div>
                        <p class="font30 greenLight mt-35 ml-7 displayNone">CARDS</p>
                </div>
                <div class="col-xs-12 font35 bolder ml-2">
                    <a class="btn font25 bolder whiteColor s-paddingLR redDarkColorHover
redHoverBG redBG hvr-buzz-out" href="{{{ URL::route('leaveGame', ['id' => '']) }}}/<?= $game-
>id ?>">Exit</a>
                </div>
            </div>
        </div>
    </div>
</div>
<script>
    var userIDFromSession = {{ $userIDFromSession }},
             boardData = {{ $boardData }},
             actionCards = {{ $actionCards }},
             gameMetadata = {{ $game->metadata }},
             synchUsers = "{{{ URL::route('updateMetadata', ['id' => $game->id]) }}}",
             removeActionCard = "{{{ URL::route('removeActionCard', ['typeID' => '']) }}}",
             validateAnswer = "{{{ URL::route('validateAnswer', ['questionID' => '',
'actionType' => '']) }}}",
             getQuestionCard = "{{{ URL::route('getQuestionCard', ['id' => $game->id, 'type'
=> '']) }}}",
             getActionCard = "{{{ URL::route('getActionCard') }}}";
</script>

<script src="/vendors/jcanvas.min.js"></script>
<script src="/vendors/gameCards.js"></script>
<script src="/vendors/usersInfo.js"></script>
<script src="/vendors/dice.js"></script>
<script src="/vendors/canvas.js"></script>
<script src="/vendors/gameGeneral.js"></script>
<script src="/vendors/question.js"></script>
<script src="/vendors/action.js"></script>

@stop
```

help.blade.php

```
@extends('layouts.layout')

@section('body')
<div class="container">
    <div class="row">
        <div class="col-md-12">
            <div class="panel panel-info mt-5">
                <div class="panel-heading fontW600 fontS2 greenBG whiteColor">
                    Info
                </div>
                <div id="info" class="panel-body fontS1-3 tableMinHeight">
                    <p>- The board of the game consist of 50 blocks and can be played with at
minimum 2
                    and maximum 4 players.</p>
                    <p>- The players starts the game with the sequence that they entered it.
                    They move on the board by rolling one dice when it's their turn.</p>
                    <p>- The game is finished after all player's reach the end, and win's the
one
                    with the highest score.</p>
                    <p>- For every block that the player moves on the board he/she win's one
point.</p>
                    <p>- On some blocks of the board there are snakes or ladders.
                    If a player finish his/her roll of the dice on these blocks, he/she have
                    to follow the ladder or the snake to the block that leads.
                    Two points will be added to his/her score if a ladder will be followed and
two points will be removed from his/her score if a snake will be followed.</p>
                    <p>- Also in most of the blocks of the game there are questions.
                    If a player stops at any of them and answer correctly, he/she
                    wins one or two points depending the difficulty of it. When the question
                    is rewarded with two points then it is represented on the block with two
questionmarks.</p>
                    <p>- At the rest of the blocks that are having on top the sign of the
cogwheel, the player
                    can win an action card that gives him/her the ability to use them during
the game to their own benefit.
                    There are nine categories of these cards as you can see below:</p>
                    <ul>
                        <li>6 Cards that let you avoid following the snake after rolling the
dice.</li>
                        <li>6 Cards that let you avoid following the ladder after rolling the
dice.</li>
                        <li>5 Cards that let you move forward 3 blocks when is your turn and
after rolling the dice, also 3 points will be added to your score.</li>
                        <li>5 Cards that let you move forward 5 blocks when is your turn and
after rolling the dice, also 5 points will be added to your score.</li>
                        <li>5 Cards that let you move forward 3 blocks when is your turn and
after rolling the dice, no points will be added to your score.</li>
                        <li>5 Cards that let you move backward 3 blocks when is your turn and
after rolling the dice, no points will be removed from your score.</li>
                        <li>5 Cards that let you move backward 5 blocks when is your turn and
after rolling the dice, no points will be removed from your score.</li>
                        <li>5 Cards that let you move backward 3 blocks when is your turn and
after rolling the dice, also 3 points will be removed to your score.</li>
                        <li>6 Cards that let you double your points when answering a question
correct.</li>
                    </ul>
                </div>
                <div class="panel-footer text-center">
                    <form class="index-buttons" action="/">
                        <input type="submit" value="Back" class="btn btn-info btn-lg">
                    </form>
                </div>
```

```
            </div>
        </div>
    </div>
</div>
@stop
```

index.blade.php

```
@extends('layouts.layout')

@section('body')

<div class="container-fluid">
  <div class="row">
    <div class="col-xs-12 text-center">
      <h1 class="animated rubberBand">
        Geeky Snake
      </h1>
    </div>
  </div>
  <div class="row">
    <div class="col-xs-12 text-center">
      <img src="images/snake.png" class="img-responsive mAuto" alt="Snake">
    </div>
  </div>
  <div class="row">
    <div class="col-xs-12 text-center mt-250-">
        <form class="index-buttons" action="search">
            <input type="submit" value="PLAY" class="btn font40 bolder whiteColor s-paddingLR
greenDarkColorHover greenHoverBG greenDarkBG hvr-buzz-out">
        </form>
<!--      <button type="button" class="btn font40 bolder whiteColor s-paddingLR
greenDarkColorHover greenHoverBG greenDarkBG hvr-buzz-out">PLAY</button>-->
    </div>
    <div class="col-xs-12 text-center mt-10-">
        <form class="index-buttons" action="help" >
            <input type="submit" value="info" aria-label="Left Align"  class="noneBorder
noneBG hvr-grow font35 blueColor">
        </form>
<!--      <button type="button" aria-label="Left Align" class="noneBorder noneBG hvr-grow">-->
<!--        <span class="glyphicon glyphicon-info-sign glyphicon-align-center font35
blueColor" aria-hidden="true"></span>-->
<!--      </button>-->
    </div>
  </div>
</div>
```

search.blade.php

```
@extends('layouts.layout')

@section('body')

@include('templates.games')
<div class="container">
    <div class="row">
        <div class="col-md-12">
            <div class="panel panel-success mt-5">
                <div class="panel-heading fontW600 fontS2 greenBG whiteColor">
                    Open Games
                </div>
                <div id="search-results" class="panel-body fontS1-3 tableMinHeight">

                </div>
                <div class="panel-footer text-center">
                    <form class="index-buttons" action="/">
                        <button id="create" class="btn btn-success btn-lg mr-
1">Create</button>
                        <input type="submit" value="Back" class="btn btn-info btn-lg">
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

{{ Form::open(['route' => ['create'], 'id' => 'create-game-player', 'style' => 'display:
none', 'class'=> 'form-horizontal whiteBlueBG m-padding']) }}
<div class="form-group">
    <label for="game-name" class="col-sm-2 control-label">Game Name</label>
        <div class="col-sm-10">
            <input id="game-name" class="form-control" maxlength="10" name="game-name"
type="text" value="">
        </div>
</div>
@unless (Session::has('user'))
<div class="form-group">
    <label for="player-name" class="col-sm-2 control-label">Player Name</label>
    <div class="col-sm-10">
        <input id="player-name" name="player-name" maxlength="6" class="form-control"
type="text" value="">
    </div>
</div>
@endunless
<div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
        <label class="radio-inline">
            <input id="non-private" class="group-private" type="radio" name="group-private"
value="public" checked>Public
        </label>
        <label class="radio-inline">
            <input id="private" class="group-private" type="radio" name="group-private"
value="private">Private
        </label>
    </div>
</div
<div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
        <input id="password" type="password" name="password" maxlength="6" class="hidden form-
control" placeholder="Type Password">
    </div>
```

```
</div>
<div class="form-group">
  <div class="col-sm-offset-2 col-sm-10 mt-5" data-toggle="tooltip" data-placement="bottom"
title="There will be 60 seconds before the game starts after the second player join's it">
        <input type="submit" value="Create" class="btn btn-default">
  </div>
</div>
</div>
{{ Form::close() }}

{{ Form::open(['route' => ['joinGame', ""], 'id' => 'join-game-player', 'style' => 'display:
none', 'class'=> 'form-horizontal whiteBlueBG m-padding'])}}

    <div id="group-player-name">
        <label for="player-name">Player Name:</label>
        <input id="player-name" name="player-name" type="text" value="" data-text="name">
    </div>
    <div id="group-password">
        <label for="new-player-join-password">Password:</label>
        <input id="new-player-join-password" name="new-player-join-password" type="password"
value="" data-text="pass" >
    </div>
    <input type="submit" value="Join">

{{ Form::close() }}

<script>

    var intervalCode,
        userExist = "{{ Session::has('user') }}",
        actionUrl = $('#join-game-player').attr('action'),
        template = $("#game-results-template"),
        gamesData = {{ $games }},
        loadOpenGames = function(data) {
            var output = _.template(template.html())({ games : data });
            $("#search-results").html(output);

            $('.join-game').click(function(e) {

                var isPrivate = $(this).data("private"),
                          url = [actionUrl, $(this).data("id")];

                $('#join-game-player').attr('action', url.join('/'));

                if (!userExist && isPrivate == 1) {
                    e.preventDefault();
                    if ($('#group-player-name').hasClass("hidden")) {
                        $('#group-player-name').removeClass("hidden");
                        $('#player-name').prop("disabled", false);
                    }
                    if ($('#group-password').hasClass("hidden")) {
                        $('#group-password').removeClass("hidden");
                        $('#new-player-join-password').prop("disabled", false);
                    }
                    $('#join-game-player').lightbox_me({
                        centered: true
                    });
                } else if(isPrivate == 1) {
                    e.preventDefault();
                    $('#group-player-name').addClass("hidden");
                    $('#player-name').prop("disabled", true);
                    $('#join-game-player').lightbox_me({
                        centered: true
                    });
                } else if(!userExist) {
                    e.preventDefault();
```

```
                    $('#group-password').addClass("hidden");
                    $('#new-player-join-password').prop("disabled", true);
                    $('#join-game-player').lightbox_me({
                        centered: true
                    });
                }
            });
        };

    $('#join-game-player').on('submit', function() {
            var inputs = $('#join-game-player').find('input[type="password"]:enabled,
input[type="text"]:enabled'),
                returnOnSubmit = true;
            $.each(inputs, function(key, value) {
                if ($(value).val() == '') {
                    var text = $(value).data("text");
                    alert("Cannot have empty " +text);

                    return returnOnSubmit = false;
                }
            });

        return returnOnSubmit;
        });

    $('#create').click(function(e) {
        $('#create-game-player').lightbox_me({
            centered: true
        });
        e.preventDefault();
    });

    loadOpenGames(gamesData);

    intervalCode = setInterval(function() {
        $.getJSON( "{{{ URL::route('search') }}}", loadOpenGames);
    }, 3000);

    $('.group-private').change(function(){
        var isPrivate = $("#private:checked"),
        password = $("#password");
        if(isPrivate.length == 1) {
            password.removeClass('hidden');
        } else {
            password.addClass('hidden');
        }
    }).change();

</script>

@stop
```

routes.php

```
<?php

/*
|--------------------------------------------------------------------------
| Application Routes
|--------------------------------------------------------------------------
|
| Here is where you can register all of the routes for an application.
| It's a breeze. Simply tell Laravel the URIs it should respond to
| and give it the Closure to execute when that URI is requested.
|
*/

Route::pattern('id', '[0-9]+');

Route::get('', 'IndexController@indexAction');
Route::get('help', 'IndexController@helpAction');

Route::get('search', ['as' => 'search', 'uses' => 'GameController@searchAction']);
Route::post('game/create', ['as' => 'create', 'uses' => 'GameController@createAction']);
Route::get('game/{id}', ['as' => 'game', 'uses' => 'GameController@playAction']);
Route::get('join/{id}', ['as' => 'join', 'uses' => 'GameController@joinAction']);
Route::post('join/{id}', ['as' => 'joinGame', 'uses' =>
'GameController@addPlayerToGameAction']);
Route::get('leave/{id}', ['as' => 'leaveGame', 'uses' => 'GameController@leaveGameAction']);
Route::get('question/{id}/{type}', ['as' => 'getQuestionCard', 'uses' =>
'GameController@getQuestionAction']);
Route::get('actionCard', ['as' => 'getActionCard', 'uses' =>
'GameController@getActionCardAction']);
Route::get('removeActionCard/{actionType}', ['as' => 'removeActionCard', 'uses' =>
'GameController@removeActionCardFromUserAction']);
Route::get('validate/{questionID}/{actionType}', ['as' => 'validateAnswer', 'uses' =>
'GameController@validateAnswerAction']);

Route::get('update/{id}', ['as' => 'updateMetadata', 'uses' =>
'GameController@updateMetadataAction']);
```

index.css

```
body {margin:0;font-family:'Jura', sans-serif;}
.mt-5 {margin-top:5%;}
.mr-1 {margin-right:1%;}
.green {color:#3c763d;}
.fontW600 {font-weight:600;}
.fontW800 {font-weight:800;}
.fontS1-3 {font-size:1.3em;}
.fontS2 {font-size:2em !important;}
.fontS3 {font-size:3em !important;}
.floatL {float:left;}
h1 {font-size:80px;margin:16px 0 0 0}
.gamePanel {background: url("/images/generalBG.jpg") center repeat;}
.playerBG {height:25%;margin:10% 0;border-radius:4px;border-top:3px solid #a0a0a0;border-
right:3px solid #a0a0a0;border-bottom:8px solid #a0a0a0;border-left:3px solid #a0a0a0;}
.background-snake {
  background:url(/images/snake.jpg) no-repeat center center;
  background-attachment:fixed;
  -webkit-background-size:cover;
  -moz-background-size:cover;
  -o-background-size:cover;
  background-size:cover;
}
.panel {opacity:0.8;}
.tableMinHeight {max-height:500px;overflow-y:scroll;}
```

main.css

```
/* GENERAL /////////////////////////////////////////////////*/
html, body, div, span, h1, h2, h3, h4, h5, h6, p, hr, a, a img, font, img, dd, dl, dt, li, ol,
ul, blockquote, fieldset, form, label, legend, table, caption, tr, th, td {
    font-size:100%;font-style:inherit;font-weight:inherit;line-height:inherit;border:0
none;background:none repeat scroll 0 0 transparent;}
html {min-height:100%;position:relative;}
* {margin:0;padding:0;}
img {border:0;}
h1 {margin:0;}
ul {padding-left:3em;}
body {font-family:'Jura', sans-serif;padding:2% 0 0 0 !important;background-color:#dcdb1d
!important;}
button,a {text-decoration:none;color:inherit;-webkit-transition:all 200ms ease-in-out;-moz-
transition:all 200ms ease-in-out;-ms-transition: all 200ms ease-in-out;-o-transition: all
200ms ease-in-out;transition: all 200ms ease-in-out;}
a:link, a:visited, a:hover, a:active {text-decoration:none;outline:none;}
a:hover {color:inherit;}

/*Overwrite Bootstrap*/
.panel-success, .panel-info {border-color: #4a8034 !important; box-sizing: border-box;}
.panel-heading {border-color: #4a8034 !important; background-image: none !important;}



/* Titles */
h1 {font-weight:600;font-size:7em;
      /* If we weren't using text-shadow, we'd set a fallback color
     and use this to set color instead
    -webkit-text-fill-color:#aec651; */
    color:#4a8034;
      /* WebKit (Safari/Chrome) Only */
      -webkit-text-stroke:1px #4a8034;
      text-shadow:4px 4px 0 #8db639,
      /* Simulated effect for Firefox and Opera
      and nice enhancement for WebKit */
      -1px -1px 0 #8db639, 1px -1px 0 #8db639, -1px  1px 0 #8db639, 1px  1px 0 #8db639;
      }

/* Font */
.bolder {font-weight:800;}
.font15 {font-size:15px;}
.font20 {font-size:20px;}
.font25 {font-size:25px;}
.font30 {font-size:30px;}
.font35 {font-size:35px;}
.font40 {font-size:40px;}
.font60 {font-size:60px;}

/* Color */
.whiteColor {color:#fff !important;}
.blueColor {color:#6593cd;}
.greenLight {color:#8eb639;}
.greenColor {color:#4a8034;}

/* Margin */
.mAuto {margin:0 auto;}
.mAutoLR {margin-left:auto !important;margin-right:auto !important;}
.mt-5 {margin-top:5%;}
.mt-5- {margin-top:-5%;}
.mt-10 {margin-top:10%;}
.mt-10- {margin-top:-10%;}
.mt-20- {margin-top:-20px;}
```

```
.mt-25- {margin-top:-25%;}
.mt-35 {margin-top:35%;}
.mt-50- {margin-top:-50px;}
.mt-100- {margin-top:-100px;}
.mt-200- {margin-top:-200px;}
.mt-250- {margin-top:-250px;}
.mt-600- {margin-top:-600px;}

.ml-2 {margin-left:2%;}
.ml-7 {margin-left:7%;}

.mb-5 {margin-bottom:5%;}

/* Padding */
.noPaddingLR {padding-left:0 !important;padding-right:0 !important;}
.m-paddingLR {padding-left:2% !important;padding-right:2% !important;}
.m-padding {padding: 2%;}

.pt-1-3 {padding-top:1.3em;}
.pt-2 {padding-top:2em;}

.pb-1 {padding-bottom:1em;}

.pl-1-4 {padding-left:1.4em;}
.pl-1- {padding-left:1em;}
.pl15 {padding-left:15px;}

/* Border */
.noneBorder {border:none !important;}
.borderB {border-bottom:2px solid #adc551;}

/* Display */
.displayBlock {display:block !important;}
.displayNone {display:none !important;}

/* Hovers */
.greenDarkColorHover:hover {color:#003209;}
.redDarkColorHover:hover {color:#640000;}
.greenHoverBG:hover {background-color:#8db639 !important;}
.redHoverBG:hover {background-color:#C5332C !important;}

/* Opacity */
.opacityPlayerNot {opacity:1 !important;}
.opacityPlayer {opacity:0.5;}

/* Float */
.floatL {float: left;}

/* Background */
.noneBG {background:none !important;}
.whiteBlueBG {background-color: #6593cd !important;}
.redBG {background-color:#640000;}
.greenLightBG {background-color:#adc551;}
.greenBG {background-color:#4a8034 !important;}
.greenDarkBG {background-color:#0d3313;}


#avatar02BG {background:url("../images/avatars/avatarBG02.png") center no-repeat;}
#avatar03BG {background:url("../images/avatars/avatarBG03.png") center no-repeat;}
#avatar04BG {background:url("../images/avatars/avatarBG04.png") center no-repeat;}
.dice-1 {width:200px;height:200px;background:url("../images/dice/d1.png") center no-repeat;}
.dice-2 {width:200px;height:200px;background:url("../images/dice/d2.png") center no-repeat;}
.dice-3 {width:200px;height:200px;background:url("../images/dice/d3.png") center no-repeat;}
.dice-4 {width:200px;height:200px;background:url("../images/dice/d4.png") center no-repeat;}
.dice-5 {width:200px;height:200px;background:url("../images/dice/d5.png") center no-repeat;}
.dice-6 {width:200px;height:200px;background:url("../images/dice/d6.png") center no-repeat;}
```

```
.cardBG {width:227px;height:250px;background:url("../images/cardsBG.png") center no-repeat;}
.actionCard {width:227px;height:250px;background:url("../images/card02.png") center no-
repeat;}
#cardOneContainer, #cardTwoContainer {width:121px;height:171px;margin:0;z-
index:1;position:relative;}
#cardOneContainer, #cardTwoContainer {perspective:1000;}
#cardOne, #cardTwo {width:100%;height:100%;transform-style:preserve-3d;transition:all 1.0s
linear;}
#cardOneContainer:hover #cardOne, #cardTwoContainer:hover #cardTwo
{transform:rotateY(180deg);}
.faceCardOne, .faceCardTwo {position:absolute;width:100%;height:100%;backface-
visibility:hidden;}
.faceCardOne.back
{color:white;text-align:center;padding:10px;display:block;transform:rotateY(180deg);box-
sizing:border-box;background:url("../images/cardEmpty01.png") center no-repeat;}
.faceCardTwo.back
{color:white;text-align:center;padding:10px;display:block;transform:rotateY(180deg);box-
sizing:border-box;background:url("../images/cardEmpty02.png") center no-repeat;}
.table-responsive, #info {max-height:550px !important;overflow-y:scroll !important;}

@media (min-width:100px) and (max-width:768px){
      .mt-50- {margin-top:0px;}
      .font35 {font-size:20px !important;}
    }

@media (min-width:768px) and (max-width:991px){
      .mt-50- {margin-top:-40px;}
      .font25 {font-size:20px !important;}
      .font35 {font-size:25px !important;}
    }


#container { text-align: center; margin: 20px; }
h2 { color: #CCC; }
a { text-decoration: none; color: #EC5C93; }

#login-players { margin-left: auto; margin-right: auto; width: 65%;}

.timer-bar-main-container {
  display: none;
  margin: 10px auto;
  width: 450px;
  height: 80px;
  z-index: 99;
  padding: 8px;
  -webkit-border-radius: 4px;
  -moz-border-radius: 4px;
  border-radius: 4px;
  color: #FFF;
  position: absolute;
  top: -webkit-calc(35% - 16px);
  top: -moz-calc(35% - 16px);
  top: calc(35% - 16px);
  left: -webkit-calc(50% - 217px);
  left: -moz-calc(50% - 217px);
  left: calc(50% - 217px);
}


.timer-bar-container {
  float: right;
  -webkit-border-radius: 20px;
  -moz-border-radius: 20px;
  border-radius: 20px;
  height: 20px;
  background: rgba(0,0,0,0.13);
```

```
  width: 100%;
  margin: 12px 0px;
  overflow: hidden;
}

.timer-bar {
  float: left;
  background: #FFF;
  height: 100%;
  -webkit-border-radius: 10px 0px 0px 10px;
  -moz-border-radius: 10px 0px 0px 10px;
  border-radius: 10px 0px 0px 10px;
}


/* COLORS */
.azure   { background: #38B1CC; }
.emerald { background: #2CB299; }
.violet  { background: #8E5D9F; }
.yellow  { background: #EFC32F; }
.red     { background: #E44C41; }

#wait-background {
  position: absolute;
  top: 0px;
  left: 0px;
  height: 100%;
  width: 100%;
  min-height: 100%;
  z-index: 5;
  display: none;
  background: black;
  opacity: 0.5;
}
```

action.js

```
$(document).bind('renderAction', function(event, delay, actionUsage) {
    actionUsage = actionUsage || 0;
    if (actionUsage == 0) {
        delay = 0;
    }
    setTimeout(function(){
        $.getJSON(window.getActionCard, function(data) {
            var output = _.template($("#action-card-template").html())({ actionData : data });
            $('#myModal').html(output).modal({'show': true});

            $('#modal-action-btn').on('click', function(event) {
                $('#action-Card').attr('src', '../images/card02.png');
                $('#myModal').modal('hide');
            });
        });
    }, delay);
});
```

canvas.js

```
var Canvas = function() {
    this.currentUserIndex = 0;
    this.activeUsers = [];
    this.associateUserDbIDJSUserIndex = [];
    this.activeUser = false;

    $('canvas').drawImage({
        source: '/images/panel.png',
        layer: true,
        x: 0, y: 0,
        width: 946,
        height: 659,
        fromCenter: false
    });
};

Canvas.prototype.addNewUser = function(userid){
    this.associateUserDbIDJSUserIndex[++this.currentUserIndex] = userid;
    this.activeUsers.push(userid);

    $('canvas').drawImage({
        source: '/images/avatars/avatar0' + this.currentUserIndex + '.png',
        layer: true,
        name: 'Avatar' + userid,
        x: 80, y: 625,
        width: 40,
        height: 40
    });
}

Canvas.prototype.removeUser = function(userid){
    $('canvas').removeLayer('Avatar' + userid);

    this.currentUserIndex--;
    //Ajax Remove User from Metadata
}

Canvas.prototype.animateUser = function(userid, fromPosition, toPosition, isMySelf,
actionUsage) {
    var diff = Math.abs(toPosition - fromPosition),
        actionUsage = actionUsage || 11;
    if (fromPosition == toPosition) {
        return;
    }

    if (fromPosition <= toPosition) {
        for (var i = fromPosition; i <= toPosition; i++) {
            $('canvas').animateLayer('Avatar' + userid, boardData[i].position);

            if(i == toPosition && isMySelf &&
window.gameMetadata.users[userid].actionCards.length == 0 && !(toPosition > 50)) {
                $(document).trigger('blockEvent', [toPosition, userid, diff, actionUsage,
'1']);
            } else if (i == toPosition && isMySelf &&
window.gameMetadata.users[userid].actionCards.length > 0 && !(toPosition > 50)) {
                $(document).trigger('askForAction', [toPosition, userid, diff]);
            }
        }
    } else if (fromPosition >= toPosition && isMySelf) {
        $('canvas').animateLayer('Avatar' + userid, boardData[toPosition].position);
        if (!(toPosition > 50)) {
            $(document).trigger('blockEvent', [toPosition, userid, diff, actionUsage, '2']);
```

```
        }
    } else if (fromPosition >= toPosition && !isMySelf) {
        if (!(toPosition > 50)) {
            $('canvas').animateLayer('Avatar' + userid, boardData[toPosition].position);
        }
    }
}
```

dice.js

```
var Dice = function() {
    return {
        randomDice : 6,

        $button : $("#btnThrowDice"),

        faces : [
            "/images/dies/d1.gif",
            "/images/dies/d2.gif",
            "/images/dies/d3.gif",
            "/images/dies/d4.gif",
            "/images/dies/d5.gif",
            "/images/dies/d6.gif"
        ],

        throwDice : function(){
            this.$button.removeClass("dice-" + this.randomDice);
            this.randomDice = Math.round(Math.random()*5) + 1;
            this.$button.addClass("dice-" + this.randomDice);
        },

        init : function (userInfo) {
            var _this = this;
            this.$button.prop('disabled', true);
            this.$button.on('click', function() {
                _this.throwDice();
                userInfo.syncUser({ 'position':  _this.randomDice, 'activeUser': true});
            });
        }
    }
};
```

gameCards.js

```
$(document).bind('blockEvent', function(event, blockPosition, userID, diff, actionUsage,
whoIsMyParrent) {
    if (whoIsMyParrent == '1') {
        diff = 0;
    } else {
        diff = diff || 0;
    }
    var animationTime = 450;

    console.log(actionUsage);

    if (boardData[blockPosition].jump !== false && actionUsage != 1 && actionUsage != 2) {
        window.userInfo.syncUser({'jumpTo':  boardData[blockPosition].jump });
        $('canvas').animateLayer('Avatar' + userID,
boardData[boardData[blockPosition].jump].position);
    } else if(boardData[blockPosition].question == 'hard') {
        $(document).trigger('renderQuestion', [(diff * animationTime), 'hard', actionUsage]);
    } else if(boardData[blockPosition].question == 'easy') {
        $(document).trigger('renderQuestion', [(diff * animationTime), 'easy', actionUsage]);
    } else if(boardData[blockPosition].actionCard == true) {
        $(document).trigger('renderAction', [(diff * animationTime), actionUsage]);
    }
});

$(document).bind('askForAction', function(event, currentPosition, userID, diff) {
    diff = diff || 0;
    var animationTime = 450;
    var availableActionCards = [];

         window.gameMetadata.users[userID].actionCards.forEach(function(actionCard) {
            var actionOnQuestionBlock = ['3', '4', '5', '6', '7', '8', '9'];
            var actionOnEmptyBlock = ['3', '4', '5', '6', '7', '8'];
            if (boardData[currentPosition].question) {
                if (actionOnQuestionBlock.indexOf(actionCard) !== -1) {
                    availableActionCards.push(actionCard);
                }
            } else if (boardData[currentPosition].jump && boardData[currentPosition].jump >
currentPosition) {
                if (actionCard == 2) {
                    availableActionCards.push(actionCard);
                }
            } else if (boardData[currentPosition].jump && boardData[currentPosition].jump <
currentPosition) {
                if (actionCard == 1) {
                    availableActionCards.push(actionCard);
                }
            } else if (boardData[currentPosition].question == false &&
boardData[currentPosition].actionCard == false && boardData[currentPosition].jump == false) {
                if (actionOnEmptyBlock.indexOf(actionCard) !== -1) {
                    availableActionCards.push(actionCard);
                }
            }
        });

    if (availableActionCards.length > 0) {
        setTimeout(function() {
            var output = _.template($("#ask-for-action-template").html())();
            $('#myModal').html(output).modal({'show': true});
            $('#no-use-action').on('click', function(event) {
                $(document).trigger('blockEvent', [currentPosition, userID, diff, 0, '1'])
            });
```

```
            $('#use-action').on('click', function(event) {
                var output = _.template($("#use-action-cards-template").html())({ actionCards
: window.actionCards, userActionCards : availableActionCards });
                $('#myModal').html(output);

                $('#apply-action').on('click', function(event) {
                    event.preventDefault();
                    var actionType = $('input:radio[name=action]:checked').val();
                    var url = window.removeActionCard + '/' + actionType;
                    $.getJSON(url)
                        .done(function() {
                            $('#myModal').modal('hide');

                            var currentPosition =
window.gameMetadata.users[userID].currentPosition, newPosition = 0;

                            if (actionType == 3) {
                                newPosition = currentPosition + 3;
                                window.userInfo.syncUser({'jumpTo':  newPosition,
actionCardValue : 3, stopExtra : 'false' });
                                canvasInstance.animateUser(userID, currentPosition,
newPosition, true, actionType);

                            } else if (actionType == 4) {
                                newPosition = currentPosition + 5;
                                window.userInfo.syncUser({'jumpTo':  newPosition,
actionCardValue : 5, stopExtra : 'false' });
                                canvasInstance.animateUser(userID, currentPosition,
newPosition, true, actionType);

                            } else if (actionType == 5) {
                                newPosition = currentPosition + 3;
                                window.userInfo.syncUser({'jumpTo':  newPosition,
actionCardValue : 0, stopExtra : 'false' });
                                canvasInstance.animateUser(userID, currentPosition,
newPosition, true, actionType);

                            } else if (actionType == 6) {
                                newPosition = currentPosition - 3;

                                window.userInfo.syncUser({'jumpTo':  newPosition,
actionCardValue : 0, stopExtra : 'false' });
                                canvasInstance.animateUser(userID, currentPosition,
newPosition, true, actionType);

                            } else if (actionType == 7) {
                                newPosition = currentPosition - 5;

                                window.userInfo.syncUser({'jumpTo':  newPosition,
actionCardValue : 0, stopExtra : 'false' });
                                canvasInstance.animateUser(userID, currentPosition,
newPosition, true, actionType);

                            } else if (actionType == 8) {
                                newPosition = currentPosition - 3;

                                window.userInfo.syncUser({'jumpTo':  newPosition,
actionCardValue : -3, stopExtra : 'false' });
                                canvasInstance.animateUser(userID, currentPosition,
newPosition, true, actionType);

                            } else {
                                $(document).trigger('blockEvent', [currentPosition, userID,
diff, actionType, 'askForAction.2']);
                            }
                    });
```

```
                });
            });
        }, diff * animationTime);
    } else {
        $(document).trigger('blockEvent', [currentPosition, userID, diff, 10,
'askForAction.3']);
    }
});
```

gameGeneral.js

```
var canvasInstance = new Canvas();
var userInfo = UsersInfo(canvasInstance);

secondsLeft = window.gameMetadata.gameStats.gameStarted ? 0 : 10 -
(window.gameMetadata.users[userIDFromSession].timeAdded -
window.gameMetadata.gameStats.gameStartedTime);

if (window.gameMetadata.gameStats.gameStarted == false) {
    $("#wait-background").show().css("height", $("#container").height());
    $("#bar-1").show();

    $('.timer-bar-percentage[data-percentage]').each(function () {
        var progress = $(this);
        var percentage = Math.ceil($(this).attr('data-percentage'));

        $({countNum: 0}).animate({countNum: percentage}, {
            duration: secondsLeft * 1000,
            easing:'linear',
            step: function() {
                var pct = '';
                pct = Math.floor(this.countNum+1);
                progress.siblings().children().css('width',pct * 6);
            },
            complete: function() {
                $("#bar-1").hide();
                $("#wait-background").hide();
                userInfo.syncUser({'gameStarted' : true});
            }
        });
    });
}
setTimeout(function() {

intervalCode = setInterval(userInfo.syncUser, 2500);
userInfo.syncUser();

$('body').mousedown(function(e){
    e.preventDefault();
})

var dice = new Dice();
dice.init(userInfo, canvasInstance);
}, secondsLeft * 1000);
```

question.js

```
$(document).bind('renderQuestion', function(event, delay, type, actionUsage) {

    setTimeout(function(){
        $.getJSON(window.getQuestionCard + '/' + type, function(data) {
            var output = _.template($("#questions-template").html())({ questionData : data });
            $('#myModal').html(output).modal({'show': true})

            var radio = $('input:radio[name=answer]');
            radio.prop('checked', false);
            $('#btnQuestionAnswer').attr('disabled', 'true');
            radio.click(function () {
                var checkval = $('input:radio[name=answer]:checked').val();
                if (checkval == '1' || checkval == '2' || checkval == '3' || checkval == '4')
{
                    $('#btnQuestionAnswer').removeAttr('disabled');
                }
            });

            $('#questionForm').submit(function(event) {
                event.preventDefault();
                var url = window.validateAnswer + '/' + $('#questionID').val() + '/' +
actionUsage;
                var checkval = $('input:radio[name=answer]:checked').val();

                $.getJSON(url, {answer :checkval})
                    .done(function(data) {
                        $('#btnQuestionAnswer').hide();
                        if (data.givenAnswer == data.rightAnswer) {
                            $('#label-' + data.givenAnswer).addClass('text-success');
                            $('#answer-result').append("Gongratulations your answer was
correct!");
                        } else {
                            $('#label-' + data.givenAnswer).addClass('text-danger');
                            $('#label-' + data.rightAnswer).addClass('text-success');
                            $('#answer-result').append("Sorry, the given answer was
incorrect.");
                        }
                        setTimeout(function() {
                            $('#myModal').modal('hide');
                        }, 2500);
                });
            });
        });
    }, delay);
});
```

usersInfo.js

```
var UsersInfo = function(canvasInstance){
    var lastUsersPosition = [],
        lastCurrentStatus = '',

        loadUsersInfo = function(data) {
            if (lastCurrentStatus == data.status) {
                return;
            }
            lastCurrentStatus = data.status;

            window.gameMetadata = data.metadata;
            window.stopExtra = data.stopExtra || true;

            _(data.metadata.users).each(function (userData, userID) {
//                if (lastUsersPosition[userID] >= 50) {
//                    return;
//                }

                var isCurrentPositionAtTheEnd       = (userData.currentPosition >= 50),
                    isCurrentPositionAtTheBegging   = (userData.currentPosition == 0),
                    isMySelf                        = (userID == userIDFromSession),
                    isUsersTurn                     = (userData.activeUser == true),
                    isUserNotExistsInGame           =
(canvasInstance.activeUsers.indexOf(userID) === -1),
                    isInSameLocation                = (lastUsersPosition[userID] ==
userData.currentPosition),
                    userFinishState                 = '',
                    isUserStateSame                 = userFinishState == userData;

                if (isUserNotExistsInGame) {
                    canvasInstance.addNewUser(userID);
                }

                if (isCurrentPositionAtTheEnd && isMySelf && !isUserStateSame) {
                    var userScore = data.metadata.users[userID].score;
                    var name = data.metadata.users[userID].name;
                    var mpeeee = _.template($("#wait-finish-game-template").html())({ score:
userScore, name : name });

                    $('#myModal').html(mpeeee).modal({'show': true});
                }

                console.log({'userID': userID, 'isMySelf' : isMySelf, 'isUsersTurn' :
isUsersTurn, 'isCurrentPositionAtTheEnd' : isCurrentPositionAtTheEnd });

                if (isMySelf && isUsersTurn && !isCurrentPositionAtTheEnd) {
                    $('#btnThrowDice').prop('disabled', false);

                } else if ((isMySelf && !isUsersTurn) || isCurrentPositionAtTheEnd) {
                    $('#btnThrowDice').prop('disabled', true);
                }

                console.info("1st", (!isCurrentPositionAtTheBegging), (!isInSameLocation),
lastCurrentStatus, lastUsersPosition, userData.currentPosition);
                if (!isCurrentPositionAtTheBegging && !isInSameLocation && window.stopExtra ==
true) {

                    var from = lastUsersPosition[userID] || 0 ,
                        to = userData.currentPosition == 50 ? 51 : userData.currentPosition;

                    canvasInstance.animateUser(userID, from, to, isMySelf);
                }
```

```
                var isReadyToJump = (boardData[userData.currentPosition].jump != false),
                    isJumpToHigherLevel = (boardData[userData.currentPosition].jump >
userData.currentPosition);

                console.info("2st", (isReadyToJump), (isJumpToHigherLevel), lastCurrentStatus,
lastUsersPosition, userData.currentPosition);

                if (isReadyToJump && isJumpToHigherLevel && isMySelf) {
                    lastUsersPosition[userID] = boardData[userData.currentPosition].jump;
                } else {
                    lastUsersPosition[userID] = userData.currentPosition;
                }

                if (data.metadata.gameStats.gameFinished) {
                    var highestScore = data.metadata.users[userID].score;
                    var name = data.metadata.users[userID].name;
                    _(data.metadata.users).each(function (userData, userID) {
                        if (highestScore < userData.score) {
                            highestScore = userData.score;
                            name = userData.name;
                        }
                    });
                    setTimeout(function() {
                        var outputt = _.template($("#win-game-template").html())({ score:
highestScore, name : name });
                        $('#myModal').html(outputt).modal({'show': true});
                    }, 400 * 5);
                }
            });
            var output = _.template($("#user-info-template").html())({ gameData :
data.metadata });
            $("#user-info-results").html(output);
    };

    return {
        $template : $("#user-info-template"),

        loadUsersInfo : loadUsersInfo,

        syncUser: function (parameter) {
            $.getJSON(window.synchUsers, (parameter || {}), loadUsersInfo);
        }
    }
}
```

cosmposer.json

```
{
      "name": "laravel/laravel",
      "description": "The Laravel Framework.",
      "keywords": ["framework", "laravel"],
      "license": "MIT",
      "require": {
        "laravel/framework": "4.2.*",
         "barryvdh/laravel-ide-helper": "1.*"
      },
    "require-dev": {
         "phpunit/phpunit": "4.2.*",
         "brianium/paratest": "dev-master",
         "mockery/mockery": "dev-master",
         "doctrine/dbal": "~2.3"
      },
      "autoload": {
        "classmap": [
             "app/commands",
             "app/controllers",
             "app/models",
             "app/database/migrations",
             "app/database/seeds",
             "app/tests/TestCase.php"
        ]
      },
      "scripts": {
        "post-install-cmd": [
                "php artisan clear-compiled",
                "php artisan optimize"
        ],
        "post-update-cmd": [
                "php artisan clear-compiled",
                "php artisan ide-helper:generate",
            "php artisan optimize"
         ],
        "post-create-project-cmd": [
                "php artisan key:generate"
        ]
      },
      "config": {
        "preferred-install": "dist"
      },
      "minimum-stability": "stable"
}
```

MAKEFILE

```
MIGRATE_NAME?=migrate_name

db.migrate:
      ./artisan migrate

db.create_migrate:
      ./artisan migrate:make $(MIGRATE_NAME)

test:
      @echo "\nStarting phpUnit...\n"
      vendor/bin/phpunit
```