ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη real-time multiplayer διαδικτυακού παιχνιδιού με υποστήριξη API για "έξυπνο" αντίπαλο παίκτη»

Του φοιτητή

Αβραμίδη Ιωάννη

Αρ. Μητρώου: 103684

Επιβλέπων

Σιδηρόπουλος Αντώνης

Επίκουρος Καθηγητής

2021

Τίτλος Δ.Ε.: Ανάπτυξη real-time multiplayer διαδικτυακού παιχνιδιού με υποστήριξη API για "έξυπνο" αντίπαλο παίκτη

Κωδικός Δ.Ε.: 21126

Ονοματεπώνυμο φοιτητή: Αβραμίδης Ιωάννης

Ονοματεπώνυμο εισηγητή: Σιδηρόπουλος Αντώνης

Ημερομηνία ανάληψης Δ.Ε: 24-02-2021

Ημερομηνία περάτωσης Δ.Ε. 07-06-2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Η εργασία αυτή είναι αφιερωμένη σε όσους/ες αγωνίζονται για ένα καλύτερο μέλλον τόσο στον κλάδο της πληροφορικής όσο και γενικότερα»

Πρόλογος

Οι βασικοί λόγοι που επέλεξα την συγκεκριμένη πτυχιακή εργασία σχετίζονται τόσο με το να αναπτύξω ένα παιχνίδι το όποιο παίζω εδώ και χρόνια ως επιτραπέζιο, όσο και με το να ασχοληθώ εκτενέστερα με την ανάπτυξη μιας διαδικτυακής εφαρμογής μέσω δημοφιλών τεχνολογιών. Το να αναπτύξω μια εφαρμογή η οποία είναι σχετική με τα ενδιαφέροντα μου, με βοήθησε στο να προσθέσω ενδιαφέρουσες λειτουργίες, στο να έχω επιπλέον κίνητρο ώστε το αποτέλεσμα να με καλύπτει και ως παίκτη και εν τέλει στο να εξοικειωθώ με πολύ χρήσιμες τεχνολογίες και πρακτικές μέσα από έναν πιο παιγνιώδη τρόπο.

Περίληψη

Η παρούσα πτυχιακή εργασία αφορά την υλοποίηση του παιχνιδιού σκορ 4 σε web περιβάλλον. Οι κανόνες του παιχνιδιού είναι απλοί. Νικητής ανακηρύσσεται όποιος καταφέρει πρώτος να τοποθετήσει 4 πούλια σε σειρά, οριζόντια, κάθετα ή διαγώνια, σε ένα ταμπλό 6 x 7 θέσεων. Κάθε παίκτης μπορεί να τοποθετήσει ένα πούλι την φορά.

Οι βασικές δυνατότητες που δίνονται στον χρήστη μέσω της εφαρμογής είναι οι εξής. Να εγγραφεί, να ξεκινήσει παιχνίδι είτε εναντίον άλλου παίκτη είτε εναντίον ενός "έξυπνου" παίκτη, να αναζητήσει αν υπάρχουν κενές θέσεις σε ήδη υπάρχοντα παιχνίδια και φυσικά να παίξει το παιχνίδι τόσο μέσα από τη διεπαφή, όσο και μέσω του REST API. Επίσης, μπορεί να δει μια λίστα με τα στατιστικά όλων των παικτών και να την ταξινομήσει βάση διαφόρων στοιχείων. Τέλος, αν βρει ένα "κρυμμένο" κουμπί που υπάρχει στην σελίδα, μπορεί πατώντας το να παίξει την καλύτερη δυνατή κίνηση εκείνη την στιγμή.

Πέρα από τις λειτουργίες που φαίνονται στον χρήστη, έχουν αναπτυχθεί και κάποιες ακόμα οι οποίες είναι βοηθητικές για την διαχείριση και την περαιτέρω ανάπτυξη της εφαρμογής. Οι πιο βασικές είναι οι εξής.

- Υλοποίηση παράλληλου thread που τρέχει μια φορά την ημέρα και διαγράφει τα παλιά μη ολοκληρωμένα παιχνίδια από την βάση δεδομένων.
- Endpoints που εμφανίζουν διάφορες μετρικές για την κατάσταση και την "υγεία" της εφαρμογής.

- Αναλυτική περιγραφή όλων των διαθέσιμων endpoint.

«Development of real time multiplayer online game using a smart opponent external API»

«Ioannis Avramidis»

Abstract

The subject of this thesis is the implementation of the “Connect 4” board game in a web application. The rules of the game are easily understandable. In order to win a game you should place 4 pieces in a row, horizontally, vertically or diagonally in a 6 x 7 board. Each player can drop one piece per turn.

The most important functionalities of the application are the following. The user can register to the app, create a game either versus another player or versus a smart opponent, find if there are existing games with available seats to join and of course to play the game both through the interface and the REST API. Also, the user will be able to find statistics information of the other players and sort the rank based on many attributes. Last but not least, somewhere in the page there is one hidden cheat button that plays the best move for the player.

Also, there are some functionalities which are not available to the players and target to help the administration and the further development of the application. Some of them are the following:

- Background worker which runs once a day and deletes all the old and uncompleted games from the database.
- Endpoints which display metrics and information about the health and the status of the application.
- Extensive documentation of the available endpoints.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Σιδηρόπουλο Αντώνη για την πολύτιμη βοήθειά του και τις εύστοχες παρατηρήσεις του.

Επίσης θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου και τους κοντινούς μου ανθρώπους που με βοήθησαν ποικιλοτρόπως στο να ανταπεξέλθω στις σπουδές μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript And XML
JVM	Java Virtual Machine
IoC	Inversion Of Control
DI	Dependency Injection
JPA	Java Persistence API
ORM	Object Relational Mapping
CRUD	Create Read Update Delete
JDBC	Java Database Connectivity
MVC	Model View Controller
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
REST	REpresentational State Transfer
API	Application Programming Interface
DTO	Data Transfer Object
HATEOAS	Hypermedia as the Engine of Application State

Κεφάλαιο 1ο: Τεχνολογίες που χρησιμοποιήθηκαν

1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει μια αναλυτική περιγραφή όλων των τεχνολογιών που χρησιμοποιήθηκαν και συνέβαλαν στο τελικό αποτέλεσμα.

1.2 HTML

Η HTML είναι η βασική γλώσσα σήμανσης για την δημιουργία ιστοσελίδων. Τα αρχεία HTML παρέχουν οδηγίες στους browsers μέσω των οποίων καθορίζεται το περιεχόμενο καθώς και ο τρόπος παρουσίασης του στις ιστοσελίδες. Συγκεκριμένα μέσω της html μπορούμε να καθορίσουμε την δομή του κειμένου (κεφαλίδες, παραγράφους), την μορφή του κειμένου (γραμματοσειρά, παραθέσεις, συνδέσμους) καθώς και να εισάγουμε φόρμες, λίστες, πίνακες, εικόνες και πλήθος άλλων στοιχείων μέσω των tag που παρέχει. Τα πιο βασικά tags που συναντώνται σε κάθε ιστοσελίδα είναι τα παρακάτω:

- <!Doctype> - Παρέχει οδηγίες σχετικά με την έκδοση της html που ακολουθεί
- <html> - Ορίζει τα πλαίσια του html κειμένου
- <head> - Παρέχει πληροφορίες σχετικές με την σελίδα (τίτλος, στύλ, πληροφορίες δεδομένων κ.α.)
- <body> - Περιλαμβάνει το κύριο σώμα της ιστοσελίδας

1.3 CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα η οποία χρησιμοποιείται για την παρουσίαση (styling) εγγράφων που είναι γραμμένα σε μια markup γλώσσα. Συνήθως η markup γλώσσα είναι η HTML, καθώς οι δυο τους και η javascript αποτελούν τον ακρογωνιαίο λίθο του world wide web. Η CSS σχεδιάστηκε για να μπορεί να είναι ευκολότερος ο διαχωρισμός του περιεχομένου και του styling μιας ιστοσελίδας καθώς αυτό μπορεί να ενισχύσει την απόδοση της και την ευελιξία στην δημιουργία της.

Η CSS μπορεί να χρησιμοποιηθεί αυτούσια, αλλά είναι πολύ συχνότερο να χρησιμοποιείται μέσω CSS framework, όπως τα tailwind, bootstrap, materialUI κ.α. τα οποία κάνουν την χρήση της απλούστερη.

1.4 Apache Freemarker

Η βιβλιοθήκη Apache Freemarker είναι μια Java βιβλιοθήκη που βοηθάει στη δημιουργία εξόδου κειμένου (ιστοσελίδες html, email templates κ.α.) με βάση κάποια πρότυπα και την επεξεργασία των δεδομένων που περνάνε σε αυτά παραμετρικά.

Τα πρότυπα (templates) είναι αποθηκευμένα στον server όπως οι στατικές σελίδες html, με την μεγάλη διαφορά ότι όταν κάποιος χρήστης επισκεφθεί την συγκεκριμένη σελίδα, ο server θα περάσει παραμετρικά όλα τα δεδομένα που χρειάζεται η συγκεκριμένη σελίδα. Το ποια δεδομένα χρειάζεται η σελίδα καθορίζεται από τον προγραμματιστή μέσω ειδικών συμβόλων (π.χ. ${variable_to_display}).

Στα freemarker templates έχουμε την δυνατότητα να ορίσουμε μεταβλητές, εντολές, να ελέγξουμε συνθήκες καθώς και πλήθος in-built μεθόδων για να διαχειριστούμε αποτελεσματικά τα δεδομένα μας. Τέλος ,τα freemarker templates βοηθάνε τον διαχωρισμό της παρουσίασης από την προγραμματιστική λογική, το όποιο είναι ιδιαίτερα βοηθητικό στην συντήρηση και την επεκτασιμότητα του κώδικα, για αυτό είναι δημοφιλές στην MVC αρχιτεκτονική.

1.5 JavaScript

Η Javascript είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου και αποτελεί μαζί με την HTML και την CSS τις κύριες τεχνολογίες του world wide web. Η γλώσσα δημιουργήθηκε από τον Brendan Eich για τον Netscape navigator browser το 1995. Η γλώσσα πήρε το συγκεκριμένο όνομα για marketing σκοπούς, καθώς η java ήταν πολύ δημοφιλής εκείνη την περίοδο. Λίγο αργότερα, το 1997 μετονομάστηκε σε ECMAscript, όνομα που κρατάει μέχρι σήμερα.

Αν και η γλώσσα αρχικά χρησιμοποιήθηκε μόνο για απλές διαδράσεις με το DOM μιας ιστοσελίδας, πλέον είναι η πιο δημοφιλής γλώσσα προγραμματισμού και μπορεί να χρησιμοποιηθεί για προγραμματισμό σε server (node.js), κατασκευή εφαρμογών σε android και ios (react-native), κατασκευή desktop εφαρμογών (electron), τεχνητή νοημοσύνη (tensorflow.js) και για αρκετούς ακόμη σκοπούς.

Η Javascript είναι μια γλώσσα με dynamic typing που υποστηρίζει τόσο object oriented όσο και functional προγραμματισμό.

1.5.1 AJAX

“Η AJAX είναι ένα σύνολο από Web development τεχνικές που χρησιμοποιούν πολλές τεχνολογίες του διαδικτύου από την πλευρά του πελάτη για να δημιουργήσουν ασύγχρονες Web εφαρμογές.

Με AJAX, οι Web εφαρμογές μπορούν να στέλνουν και να ανακτούν δεδομένα από έναν διακομιστή (server) ασύγχρονα (τρέχοντας στο παρασκήνιο), χωρίς να παρεμβαίνουν στην εμφάνιση και τη συμπεριφορά της υπάρχουσας σελίδας. Με την αποσύνδεση του επιπέδου των δεδομένων που έχουν την δυνατότητα αλλαγής από το επίπεδο παρουσίασης της σελίδας, η AJAX επιτρέπει σε Web σελίδες, και κατ' επέκταση σε Web εφαρμογές, να αλλάζουν το περιεχόμενο τους δυναμικά, χωρίς να χρειάζεται να φορτωθεί εκ νέου ολόκληρη η σελίδα.

Στην πράξη, οι σύγχρονες εφαρμογές συνήθως χρησιμοποιούν JSON, αντί για XML, λόγω των πλεονεκτημάτων του JSON που υπάρχουν εκ φυσικού στην JavaScript.”

1.6 Java

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού υψηλού επιπέδου. Αναπτύχθηκε από την εταιρεία Sun Microsystems το 1995 που στην συνέχεια αποκτήθηκε από την Oracle, η οποία είναι υπεύθυνη για την συντήρηση και την ανάπτυξη της μέχρι και σήμερα.

Το κύριο χαρακτηριστικό της, το οποίο είναι και υπεύθυνο για τα πιο πολλά πλεονεκτήματά της είναι ότι ο κώδικας που γράφεται σε Java μεταγλωττίζεται σε bytecode και αυτό με την σειρά του εκτελείται από το Java Virtual Machine, ένα runtime environment αποκλειστικό για java εφαρμογές που αλληλεπιδρά με το εκάστοτε λειτουργικό σύστημα πάνω στο οποίο τρέχει.

Τα βασικά πλεονεκτήματα που πηγάζουν από αυτό είναι τα εξής:

- WORA (Write Once, Run Anywhere). Τα προγράμματα που έχουν γραφτεί σε Java έχουν την δυνατότητα να τρέχουν σε οποιοδήποτε λειτουργικό σύστημα, χωρίς να απαιτείται κάποια αλλαγή στον κώδικα ή να επαναμεταγλωττιστούν. Αυτό κατορθώνεται λόγω των διαφορετικών υλοποιήσεων JVM ανά λειτουργικό σύστημα. Για τον παραπάνω λόγο υπάρχουν διαφορετικά JVM για Linux, MacOS, Windows, Solaris, Android etc.

- Ασφάλεια. Το JVM, το περιβάλλον στο οποίο τρέχει η εφαρμογή, έχει φτιαχτεί με τέτοιο τρόπο ώστε να προστατεύει σε μεγάλο βαθμό από πιθανές ευπάθειες.

- Το JVM είναι υπεύθυνο για την διαχείριση της μνήμης, των exceptions και πολλών ακόμα απαραίτητων εργασίων για να τρέξει σωστά ένα πρόγραμμα. Αυτό προσφέρει ένα abstraction layer σε σχέση με τις low-level γλώσσες προγραμματισμού, το οποίο διευκολύνει σημαντικά τον προγραμματιστή.

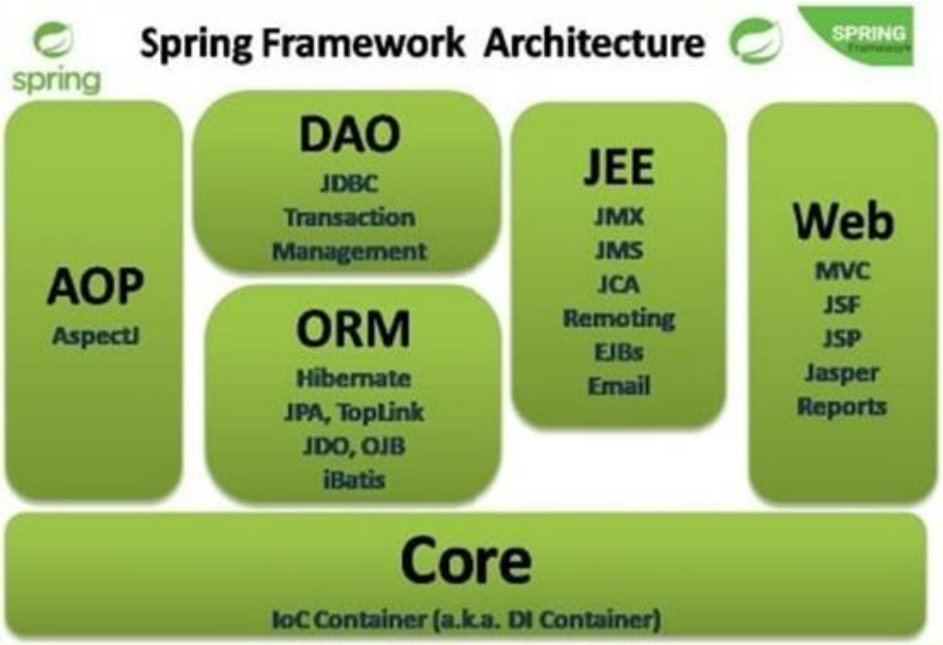
- Η Java, ως γλώσσα που χρειάζεται μεταγλώττιση, παρέχει έναν έλεγχο σε σχέση με λάθη που μπορεί να προκύψουν κατά την ανάπτυξη ενός προγράμματος, π.χ. από τυπογραφικά λάθη μέχρι ανάθεση μη συμβατών τύπων μεταβλητών. Πέραν αυτού, ο μεταγλωττιστής πραγματοποιεί βελτιστοποιήσεις στον κώδικα που δέχεται με αποτέλεσμα το τελικό αρχείο να είναι πολλές φορές πιο γρήγορα εκτελέσιμο.

Εκτός από τα παραπάνω πλεονεκτήματα η Java ως δημοφιλής γλώσσα, έχει πολύ μεγάλη εξέλιξη παρέχοντας συνεχώς νέες λειτουργίες και υιοθετώντας πρακτικές που υπάρχουν κυρίως στον functional προγραμματισμό.

1.7 Spring framework

1.7.1 Εισαγωγή

Στο παρόν κεφάλαιο θα περιγράψουμε τα βασικά χαρακτηριστικά του Spring, του πιο διαδεδομένου Java framework για ανάπτυξη enterprise εφαρμογών και θα αναλύσουμε κάποια από τα πολλά modules από τα οποία αποτελείται. Το Spring είναι ένα open source framework που κυκλοφόρησε το 2003 και από τότε έχει αναπτυχθεί τρομερά λόγω των δυνατοτήτων που προσφέρει.



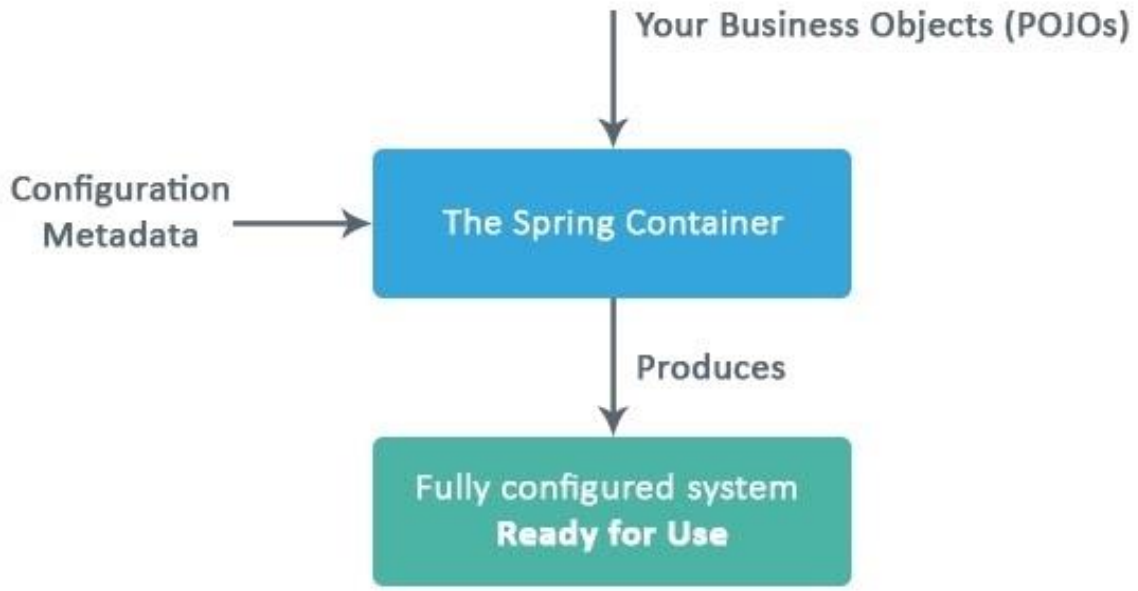
Σχήμα 1.1: Τα βασικότερα modules του Spring

1.7.2 Spring Core

Το βασικό χαρακτηριστικό του Spring πάνω στο οποίο βασίζονται πολλές λειτουργίες του είναι ο IoC container ή αλλιώς DI container. Για να γίνουν πιο κατανοητές οι προαναφερθείσες έννοιες, θα πρέπει πρώτα να καταλάβουμε τι είναι το DI. Το Dependency Injection είναι η διαδικασία όπου τα αντικείμενα λαμβάνουν παθητικά τις εξαρτησεις (dependencies) που έχουν σε άλλα αντικείμενα με τα οποία αλληλεπιδρούν.

Ο DI container, είναι υπεύθυνος για την αυτή την διαδικασία. Δημιουργεί, συνδέει μεταξύ τους και διαχειρίζεται όλα τα αντικείμενα, με τελικό στόχο να τροφοδοτήσει την εφαρμογή με όποιο αντικείμενο χρειαστεί. Έτσι, ο προγραμματιστής δεν χρειάζεται να αρχικοποιεί ένα αντικείμενο όταν το χρειάζεται μέσα σε κάποια κλάση, το δηλώνει (μέσω διαφόρων τρόπων που παρέχει το Spring) και το λαμβάνει όταν το χρειαστεί. Ακολουθώντας αυτό το σχεδιαστικό πρότυπο, το Spring:

- Παρέχει πιο αποδοτική διαχείριση των αντικειμένων.
- Βοηθάει στην συντήρηση, στην επεκτασιμότητα του κώδικα και στο να είναι πιο κατανοητός και ευανάγνωστος.
- Βοηθάει στην διαδικασία ελέγχου του κώδικα (unit και integration testing).



Σχήμα 1.2: Η λειτουργία του IoC container

1.7.3 Spring Data

Ο βασικός στόχος του Spring Data module είναι να προσφέρει έναν εύκολα διαχειρίσιμο και αξιόπιστο τρόπο διαχείρισης, πρόσβασης και επικοινωνίας με βάσεις δεδομένων. Πιο συγκεκριμένα, το συγκεκριμένο module είναι μια ομπρέλα κάτω από την οποία υπάρχουν πολλά submodules που ειδικεύονται σε διαφορετικούς τύπους βάσεων δεδομένων (σχεσιακές, μη-σχεσιακές, γράφου) και σε διαφορετικές υλοποιήσεις τους (MySQL, PostgreSQL, MongoDB κ.α). Παρακάτω θα αναλύσουμε το module που χρησιμοποιήθηκε σε αυτήν την εφαρμογή.

1.7.3.1 Spring Data JPA

Το JPA είναι ένα σύνολο προδιαγραφών σχετικά με το πως θα πρέπει να πραγματοποιείται η πρόσβαση, η αποθήκευση και η διαχείριση δεδομένων μεταξύ των Java αντικειμένων και των πινάκων μιας σχεσιακής βάσης δεδομένων. Θεωρείται η κύρια και πιο αξιόπιστη προσέγγιση για ένα ORM σε Java. Το Spring Data JPA είναι μια υλοποίηση των προδιαγραφών αυτών, προσθέτοντας ένα επιπλέον επίπεδο αφαιρετικότητας για να διευκολύνει τον προγραμματιστή. Στην ουσία είναι ένα πολύ καλά δομημένο και με πολλές λειτουργίες ORM framework.

Η κύρια λειτουργία ενός ORM εργαλείου είναι να αντιστοιχίζει κλάσεις με πίνακες μιας σχεσιακής βάσης δεδομένων. Αυτό μπορούμε να το πετύχουμε δηλώνοντας κλάσεις με συγκεκριμένα annotations και διευκρινίζοντας ποια πεδία της κλάσης θα αντιστοιχιστούν με ποιες στήλες της βάσης. Με αυτόν τον τρόπο δίνεται η δυνατότητα στον προγραμματιστή να εστιάσει στο να αναπτύξει την λογική διαχείρισης των αντικειμένων στην εφαρμογή του και να αφήσει στο ORM εργαλείο την δουλειά του πως αυτό θα μετατραπεί σε SQL.

Μια ακόμη σημαντική δυνατότητα ενός ORM που προκύπτει από τα παραπάνω είναι οι "έτοιμες" μέθοδοι που προσφέρει σε σχέση με βασικές CRUD λειτουργίες. Συγκεκριμένα υπάρχουν έτοιμες μέθοδοι που μπορεί να χρησιμοποιήσει ο προγραμματιστής για να αποθηκεύσει, να αναζητήσει και να τροποποιήσει αντικείμενα, όπως επίσης για να ταξινομήσει και να σελιδοποιήσει λίστες αντικειμένων.

Σε περίπτωση που δεν υπάρχει έτοιμη η μέθοδος που θέλει να χρησιμοποιήσει ο προγραμματιστής, τις περισσότερες φορές μπορεί να την δημιουργήσει γράφοντας απλά στα αγγλικά την επιθυμητή λειτουργία της μεθόδου του ακολουθώντας κάποιες οδηγίες στο πως θα την γράψει, οι οποίες ορίζονται από το JPA. Τα πλεονεκτήματα αυτής της δυνατότητας είναι πολλά κυρίως όσον αφορά την ευκολία του να αναπτύξεις και να συντηρήσεις κατανοητό και ευανάγνωστο κώδικα.

Τέλος μια ακόμη ευκολία που προσφέρει ένα ORM, είναι οι αυτοματοποιημένοι τρόποι για να πραγματοποιήσεις εύκολες, αξιόπιστες και αποτελεσματικές συνδέσεις με την βάση δεδομένων, όπως επίσης και να διαχειριστείς πιο σύνθετα ζητήματα (π.χ. transactions). Στην εικόνα που ακολουθεί γίνονται ορατά πολλά από τα πλεονεκτήματα που προσφέρει ένα ORM.

```
public Car getById(String id) {
     Connection con = null;
     PreparedStatement stmt = null;
     ResultSet rs = null;
     try { String sql = "select * from CAR where ID = ?";
          con =
DriverManager.getConnection("localhost:3306/cars");
          stmt = con.prepareStatement(sql);
          stmt.setString(1, id);
          rs = stmt.executeQuery();
          if(rs.next()) {
               Car car = new Car();
               car.setMake(rs.getString(1));
               return car;
          } else {
               return null;
          }
     } catch (SQLException e) {
          e.printStackTrace();
     } finally {
          try {
               if(rs != null && !rs.isClosed()) {
               rs.close();
               }
          } catch (Exception e) {
     } } return null; }
```

```
public Car findCar(String id) {
  return getEntityManager().find(Car.class,
id); }
```

Σχήμα 1.3: Υλοποίηση query με JDBC vs JPA

1.7.4 Spring MVC

Το spring MVC βασίζεται πάνω στην αρχιτεκτονική Model View Controller.

- Το Model έχει να κάνει με την υλοποίηση της βασικής λογικής μιας εφαρμογής και της επικοινωνίας με την βάση δεδομένων.

- Το View σχετίζεται με αυτό που εν τέλει θα δει ο χρήστης. Στην ουσία περιέχει την HTML η οποία θα ανανεωθεί με τα δεδομένα του model μέσω της κοινής επικοινωνίας τους με τον controller.

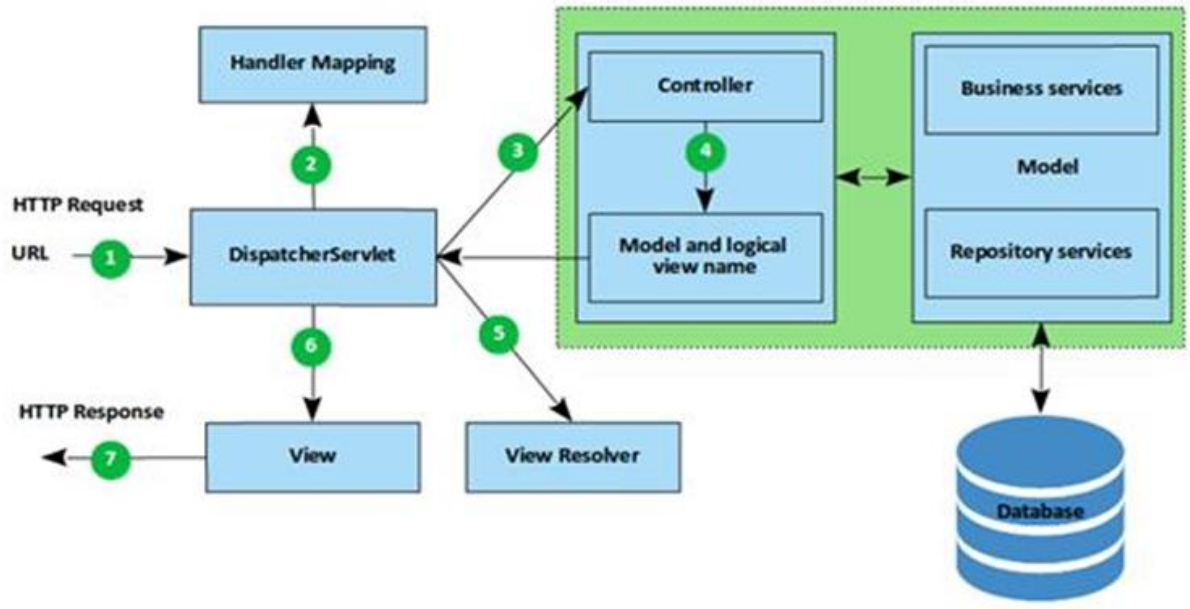
- Ο Controller θα μπορούσαμε να πούμε ότι είναι ο "εγκέφαλος" μιας MVC εφαρμογής. Όταν δεχθεί ένα request από τον χρήση, ζητάει από το model τα data που χρειάζεται για να τα μεταβιβάσει στο view για να απεικονίσει το τελικό αποτέλεσμα στον χρήστη.

Τα οφέλη της MVC αρχιτεκτονικής είναι πολλά και τα περισσότερα από αυτά προκύπτουν από τον σαφή διαχωρισμό των αρμοδιοτήτων και των ευθυνών του κάθε επιπέδου της εφαρμογής (Separation of concerns).

Τα ξεκάθαρα διαχωρισμένα επίπεδα, είναι βοηθητικά ως προς την δυνατότητα συντήρησης και επέκτασης της εφαρμογής, δηλαδή την ευκολία με την οποία μπορούμε να προσθέσουμε νέες λειτουργίες ή να λύσουμε προβλήματα που μπορεί να προκύψουν.

Ένα επιπλέον πλεονέκτημα των διαχωρισμένων επιπέδων είναι η ευκολία στο να ελεγχθεί ο κώδικας της εφαρμογής μέσω αυτοματοποιημένων διαδικασιών ελέγχου(unit test, integration test).

Για να γίνει πιο ξεκάθαρος ο τρόπος λειτουργίας του Spring MVC παραθέτω την παρακάτω εικόνα και από κάτω θα γίνει πιο εκτενής αναφορά για την "διαδρομή" που ακολουθεί ένα HTTP request.



Σχήμα 1.4: Ο τρόπος λειτουργίας του Spring MVC

1. Η διαδικασία ξεκινάει όταν ο χρήστης στείλει ένα HTTP request σε ένα συγκεκριμένο URL. Το DispatcherServlet είναι το πρώτο κομμάτι της εφαρμογής που λαμβάνει το request.

2. Το DispatcherServlet αποφασίζει ποιος είναι ο υπεύθυνος controller για την διαχείριση του request μέσω του Handler Mapping και των URL που ορίζει ο προγραμματιστής σε κάθε controller. Για παράδειγμα βάζοντας το annotation GetMapping("api/players/statistics") πάνω από την μέθοδο που επιστρέφει τα στατιστικά των παικτών, ορίζουμε ότι θα "ακούει" στο συγκεκριμένο URL.

3. Το DispatcherServlet καλεί τον υπεύθυνο controller.

4. Καλείται η κατάλληλη μέθοδος του controller, η οποία κάνει όλες τις ενέργειες που χρειάζονται και επιστρέφει το model που περιέχει τα επιθυμητά δεδομένα και το όνομα του template που ο προγραμματιστής θέλει να ενταχθούν.

5. Το DispatcherServlet με την βοήθεια του ViewResolver αποφασίζουν ποιο θα είναι το τελικό template στο οποίο θα ενταχθούν τα δεδομένα του model.

6. Το DispatcherServlet παρέχει τα δεδομένα στο επιλεγμένο view template.

7. Επιστρέφεται στον χρήστη ως response το template έχοντας όλα τα απαιτούμενα δεδομένα.

1.7.5 Spring Security

Το Spring security είναι το module του Spring που "ειδικεύεται" στο να κάνει μια Java εφαρμογή πιο ασφαλή και κυρίως στο να παρέχει εύκολους και ενδεδειγμένους τρόπους υλοποίησης λειτουργιών αυθεντικοποίησης (authentication) και εξουσιοδότησης (authorization).

Ο όρος αυθεντικοποίηση χρησιμοποιείται για να ορίσει την διαδικασία μέσω της οποίας ένας χρήστης ή μια εφαρμογή πιστοποιεί ότι είναι αυτός/ή που ισχυρίζεται. Συνήθως αυτό υλοποιείται δίνοντας το όνομα χρήστη και έναν κωδικό.

Ο όρος εξουσιοδότηση χρησιμοποιείται για να ορίσει την διαδικασία μέσω της οποίας αποφασίζεται αν ένας πιστοποιημένος χρήστης έχει δικαίωμα πρόσβασης σε συγκεκριμένους πόρους ή όχι. Συνήθως αυτό υλοποιείται αναθέτοντας ρόλους σε χρήστες και ορίζοντας τι δικαιώματα έχει ο κάθε ρόλος.

Η λειτουργία του Spring security βασίζεται πάνω στα servlet filters, δηλαδή στην τοποθέτηση φίλτρων πριν από οποιοδήποτε servlet ή σελίδα επιθυμούμε. Για την ακρίβεια μέσω του Spring security μπορούμε να δημιουργήσουμε μια αλυσίδα από φίλτρα στα οποία θα ορίσουμε τα endpoint της εφαρμογής που θέλουμε να ασφαλίσουμε, σε ποιους θα είναι προσβάσιμο κάθε endpoint και πολλές ακόμα παραμέτρους σχετικές με την ασφάλεια της εφαρμογής.

Με αυτόν τον τρόπο επιτυγχάνεται το να ανατεθεί η ασφάλεια της εφαρμογής σε ένα επίπεδο που βρίσκεται ένα βήμα πριν από τα servlets. Έτσι η υλοποίηση της λογικής "ασφάλισης" γίνεται πιο αποδοτική, πιο ασφαλής και πιο επαναχρησιμοποιήσιμη.
Ακριβώς από κάτω παρατίθενται 2 στιγμιότυπα οθόνης για την καλύτερη κατανόηση των λειτουργιών που αναλύσαμε.

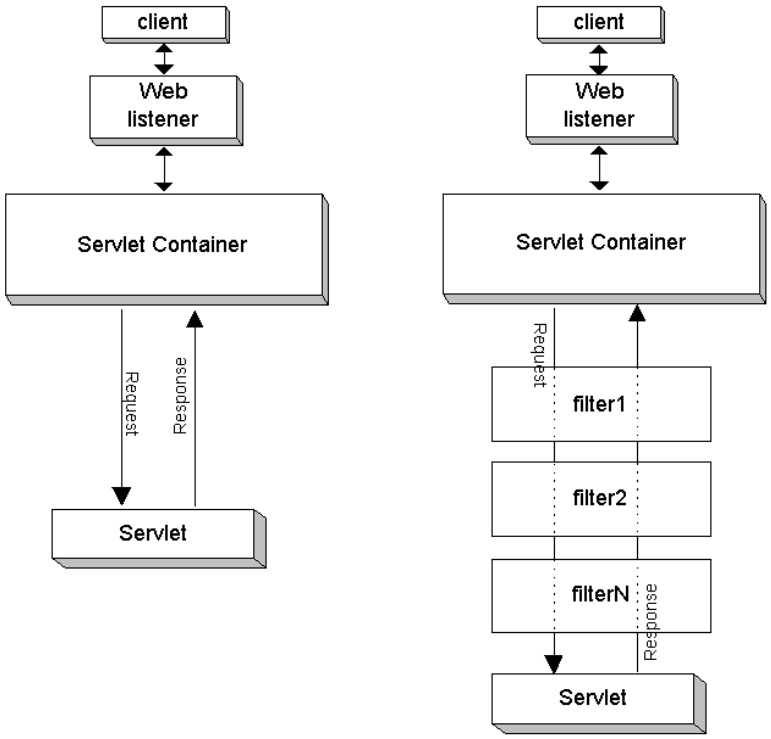
Το πρώτο απεικονίζει μια κλάση η οποία είναι υπεύθυνη για την παραμετροποίηση της ασφάλειας της εφαρμογής. Συγκεκριμένα ορίζει σε ποια endpoint θα έχουν πρόσβαση μόνο αυθεντικοποιημένοι χρήστες και με ποιον τρόπο θα γίνει η αυθεντικοποίηση.

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .authorizeRequests()
                .antMatchers( ...antPatterns: "/actuator", "/actuator/*").authenticated()
                .anyRequest().permitAll()
                .and().formLogin()
                .and().csrf().disable();
    }
}
```

Σχήμα 1.5: Παράδειγμα υλοποίησης security configuration μέσω Spring

Το δεύτερο απεικονίζει τον τρόπο που λειτουργούν τα servlet filters και κατ' επέκταση και τα chain filters του Spring security.



Σχήμα 1.6: Το επίπεδο των servlet filter

1.7.6 Spring Boot

Το Spring boot είναι ένα project που βασίζεται εξ' ολοκλήρου στο Spring με κάποια επιπλέον χαρακτηριστικά τα οποία το έχουν βοηθήσει να καθιερωθεί στην κοινότητα των Java προγραμματιστών. Ο κύριος προσανατολισμός του είναι να παρέχει λειτουργίες που θα διευκολύνουν την ανάπτυξη διαδικτυακών εφαρμογών και θα απαλλάξουν όσο γίνεται τους προγραμματιστές από τις "περιττές" ρυθμίσεις που χρειάζεται μια εφαρμογή για να στηθεί, έτσι ώστε να είναι πιο επικεντρωμένοι στην υλοποίηση της λογικής της εφαρμογής.

Το κύριο πλεονέκτημα του Spring boot είναι ότι αυτοματοποίησε την συντριπτική πλειοψηφία των ρυθμίσεων που χρειαζόταν μια Spring εφαρμογή για να "τρέξει", στηριζόμενο στο convention - over - configuration. Ο όρος αυτός χρησιμοποιείται για να περιγράψει την λογική του Spring boot να βασίσει τις προεπιλεγμένες ρυθμίσεις του σε αυτές που ήταν αναγκαίες στις περισσότερες περιπτώσεις ανάπτυξης λογισμικού. Πέραν αυτού οι επιπλέον παραμετροποιήσεις πραγματοποιούνται μέσω κώδικα Java ή ακόμα και μέσα από property αρχεία και δεν είναι αναγκαστική η χρήση xml αρχείων, τα οποία είναι πιο δυσανάγνωστα.

Εκτός του κύριου πλεονεκτήματος υπάρχουν κάποιες πολύ σημαντικές λειτουργίες που καλό είναι να αναφερθούν:

- Παρέχει ενσωματωμένο server (π.χ. tomcat ή jetty) και απλοποιεί την διαδικασία του deployment.
- Απλοποιεί την διαχείριση των dependencies (εξαρτήσεων) σε βιβλιοθήκες και framework.
- Παρέχει ένα command line εργαλείο για να επιταχύνει την όλη διαδικασία.

- Παρέχει το Spring Boot Actuator, το οποίο έχει μια σειρά από λειτουργίες για production-ready εφαρμογές, όπως μετρικές, έλεγχος της "υγείας" της εφαρμογής και εξωτερική παραμετροποίηση.

1.7.7 Spring Boot Actuator

Το Spring Boot Actuator αποτελεί ένα τμήμα του Spring Boot. Ο βασικός του στόχος είναι να βοηθήσει στον έλεγχο και την διαχείριση εφαρμογών που βρίσκονται σε production περιβάλλον.

Παρέχει μια σειρά από πληροφορίες, μέσω endpoint, που έχουν να κάνουν με την "εσωτερική" κατάσταση του περιβάλλοντος της εφαρμογής όπως:

- Εμφάνιση των κλάσεων τις οποίες διαχειρίζεται το Spring.
- Εμφάνιση των διαθέσιμων μεταβλητών περιβάλλοντος.
- Εμφάνιση των πρόσφατων HTTP request.
- Εμφάνιση πληροφοριών σχετικές με την μνήμη του συστήματος
- Εμφάνιση πληροφοριών και παραμετροποίηση των logger της εφαρμογής.
- Εμφάνιση πληροφοριών σχετικές με αλλαγές στην βάση δεδομένων.
- Εμφάνιση πληροφοριών σχετικά με τις προγραμματισμένες παράλληλες εργασίες της εφαρμογής.
- Εμφάνιση πληροφοριών σχετικές με τα session των χρηστών.
- Εμφάνιση πληροφοριών για την υγεία της εφαρμογής.
- Έλεγχος έναρξης και τερματισμού λειτουργίας της εφαρμογής.

1.8 Maven

Το maven είναι ένα εργαλείο αυτοματοποίησης build (χτισίματος) κυρίως προγραμμάτων Java το οποίο κατασκευάζεται και συντηρείται από την Apache software foundation. Οι δύο βασικές λειτουργίες του είναι οι εξής:

- Περιγράφει και διαχειρίζεται τα dependencies (εξαρτήσεις) που έχει η εφαρμογή μας σε frameworks και βιβλιοθήκες που χρησιμοποιεί.

- Περιγράφει και ορίζει διάφορα χαρακτηριστικά και στάδια σχετικά με το χτίσιμο μιας εφαρμογής. Παραδείγματος χάρη, μέσω του maven, μπορούμε να ορίσουμε εύκολα την version της εφαρμογής μας, να ορίσουμε και να αυτοματοποιήσουμε λειτουργίες όπως η διασφάλιση της ποιότητας της εφαρμογής, τρέχοντας τα unit tests και τα integration tests, ή όπως το build και το deploy της εφαρμογής.

Συγκεκριμένα οι φάσεις που ορίζονται από το maven είναι οι εξής:

- validate - επικυρώνει ότι είναι διαθέσιμες όλες οι πληροφορίες που χρειάζεται η εφαρμογή.
- compile - μεταγλωττίζει τον πηγαίο κώδικα της εφαρμογής.

- test - ελέγχει τον μεταγλωττισμένο πηγαίο κώδικα τρέχοντας τα unit tests της εφαρμογή.
- package - πακετάρει τον πηγαίο κώδικα της εφαρμογής σε αρχεία τύπου JAR κ.α.
- verify - τρέχει τα integration tests της εφαρμογής για να επιβεβαιώσει την σωστή λειτουργία της.
- install - εγκαθιστά την εφαρμογή τοπικά για να μπορεί να χρησιμοποιηθεί και ως dependency από άλλες εφαρμογές
- deploy - εγκαθιστά τα παραγόμενα αρχεία της εφαρμογής σε κάποιο αποθετήριο για να μπορεί να διαμοιραστεί με άλλους προγραμματιστές ή να "τρέξει" σε κάποιον server.

Το βασικό συστατικό του maven είναι ένα xml αρχείο με την ονομασία "pom.xml" μέσα στο οποίο ορίζονται όλες οι εξαρτήσεις της εφαρμογής και όλα τα δεδομένα που χρειάζεται για να αυτοματοποιήσει και να ορίσει την ευρύτερη διαδικασία του build.

1.9 PostgreSQL

Η PostgreSQL αποτελεί μια σχεσιακή βάση δεδομένων.
Ως σχεσιακή βάση δεδομένων εννοούμε τη συλλογή δεδομένων οργανωμένη σε συσχετιζόμενους πίνακες που παρέχουν ταυτόχρονα διάφορους μηχανισμούς για τη διαχείριση των δεδομένων (ανάγνωση, εγγραφή κοκ). Ως απώτερο σκοπό, μια βάση δεδομένων, έχει την οργανωμένη αποθήκευση αλλά και εξαγωγή της πληροφορίας ανάλογα με τα ερωτήματα που τίθενται.

Στην περίπτωση της PostgreSQL έχουμε να κάνουμε με μια ανοικτού κώδικα σχεσιακή βάση που αναπτύσσεται και τρέχει πάνω από δύο δεκαετίες. Επιπρόσθετα, η PostgreSQL τρέχει σε όλα τα βασικά λειτουργικά συστήματα (Linux, Unix, Windows) και περιλαμβάνει τους περισσότερους SQL92 και SQL99 τύπους δεδομένων, καθώς και την αποθήκευση μεγάλων binary όπως βίντεο και ήχων.

Προερχόμενη, από το σχέδιο POSTGRES, η PostgreSQL πήρε τη σταθερή της, αρχική, μορφή και όνομα, τον Ιούλιο του 1996. Σήμερα διαθέτει περιβάλλοντα προγραμματισμού για διάφορες γλώσσες καθώς και υποστήριξη για την πλατφόρμα .NET και το πρότυπο ODBC. Τέλος, η Διαχείριση της βάσης δεδομένων γίνεται μέσω του εργαλείου pgAdmin αλλά και με τη χρήση εφαρμογών τρίτων όπως PgAccess, PhpPgAdmin, WinSQL.

1.10 GitHub

Το GitHub Inc είναι ένας δημοφιλής πάροχος διαδικτυακής φιλοξενίας για ανάπτυξη λογισμικού και έλεγχο έκδοσης χρησιμοποιώντας το Git. Αναλυτικότερα, το GitHub προσφέρει τη λειτουργική διαχείριση κατανεμημένων εκδόσεων και διαχείρισης πηγαίου κώδικα (SCM) του Git, καθώς και τις δικές του δυνατότητες όπως παρακολούθηση σφαλμάτων , έλεγχο πρόσβασης, διαχείριση εργασιών, αιτήματα λειτουργιών, συνεχή ενσωμάτωση και wiki για κάθε έργο.

Οι βασικές υπηρεσίες του GitHub είναι δωρεάν ενώ παρέχει και μια γκάμα από υπηρεσίες για επαγγελματικούς σκοπούς επί πληρωμή. Από τον Ιανουάριο του 2020, το GitHub αναφέρει ότι έχει πάνω από 40 εκατομμύρια χρήστες και περισσότερα από 190 εκατομμύρια αποθετήρια ενώ το από το 2018 αποτελεί θυγατρική εταιρεία της Microsoft. Τέλος, το GitHub παρέχει τη δυνατότητα σύνδεσης

στα δημοφιλέστερα περιβάλλοντα ανάπτυξης λογισμικών ενώ παράλληλα από το 2017 έχει κυκλοφορήσει και desktop εφαρμογή.

1.11 IntelliJ IDEA

Το IntelliJ IDEA είναι ένα δημοφιλές και ολοκληρωμένο περιβάλλον για την ανάπτυξη λογισμικών γραμμένο σε JAVA από την εταιρεία JetBrains. Ξεκινώντας τον Ιανουάριο 2001, με τη version 1.0, αποτέλεσε έναν από τους πρώτους JAVA IDE με προηγμένες δυνατότητες για αναδιαμόρφωση και πλοήγηση κώδικα.

Σήμερα, το λογισμικό IntelliJ IDEA βρίσκεται στη version 2021.1.1 και διανέμεται σε 2 εκδόσεις την εμπορική Ultimate και την Community. Η Ultimate έκδοση παρέχει στον χρήστη επιπλέον εργαλεία όπως :

- Υποστήριξη για μια σειρά από JAVA Frameworks (Spring, Jakarta EE, Helidon κ.α.).
- Εκτενή υποστήριξη για ενέργειες σχετικές με Βάσεις Δεδομένων.
- Υποστήριξη για την γλώσσα JavaScript και για διάφορα JavaScript Framework.
- Υποστήριξη σχετικά με το OpenAPI Specification.

Το IntelliJ IDEA παρέχει στον χρήστη, και στις 2 εκδόσεις, μια σειρά από εργαλεία και επιλογές που καθιστούν την ανάπτυξη λογισμικών πιο εύκολη και λειτουργική. Αρχικά, παρέχει μια γκάμα από coding assistance λειτουργιών όπως την “αυτόματη” ολοκλήρωση κώδικα, αφότου έχει καταλάβει το περιεχόμενο, τον εντοπισμό σφαλμάτων και την εμφάνιση πιθανών λύσεων στον χρήστη που τις περισσότερες φορές είναι αρκετά εύστοχες.

Παράλληλα, πέρα από τα δεκάδες plug-ins για κάθε χρήση (debugger, AI, emulators κ.α.), το IntelliJ IDEA παρέχει μια σειρά από ενσωματωμένα εργαλεία για build και packaging (gradle, maven κ.α.) όπως και για διάφορα συστήματα ελέγχου εκδόσεων (Git, Mercurial κ.α.).

1.12 Heroku

Το Heroku είναι μια πλατφόρμα ως υπηρεσία (PaaS) που έχει ως βασικό στόχο να παρέχει υπηρεσίες σχετικές με το “ανέβασμα”, την επιτήρηση και την διαχείριση εφαρμογών.
“Ο όρος πλατφόρμα ως υπηρεσία χρησιμοποιείται για να ορίσει έναν τρόπο ενοικίασης υλικού, λειτουργικών συστημάτων, αποθηκευτικού χώρου και δικτυακής υποδομής μέσω του Διαδικτύου. Αυτό το είδος υπηρεσιών επιτρέπει στο πελάτη να νοικιάζει εικονικοποιημένους (Virtualized) διακομιστές και συνεργαζόμενες υπηρεσίες με σκοπό να φιλοξενήσουν υπάρχουσες εφαρμογές ή να αναπτύσσουν και να δοκιμάσουν νέες.”
Το Heroku είναι ιδιαίτερα δημοφιλές για τους εξής λόγους:

- Απλοποιεί απίστευτα την διαδικασία “ανεβάσματος” μιας εφαρμογής.
- Υποστηρίζει εφαρμογές που έχουν αναπτυχθεί σχεδόν από όλες τις γλώσσες προγραμματισμού.
- Παρέχει τρόπους ώστε ο προγραμματιστής να μπορεί εύκολα να κλιμακώσει (scale) την εφαρμογή.
- Χρησιμοποιείται από πολύ κόσμο και υπάρχει πλήθος οδηγών για τις λειτουργίες του.

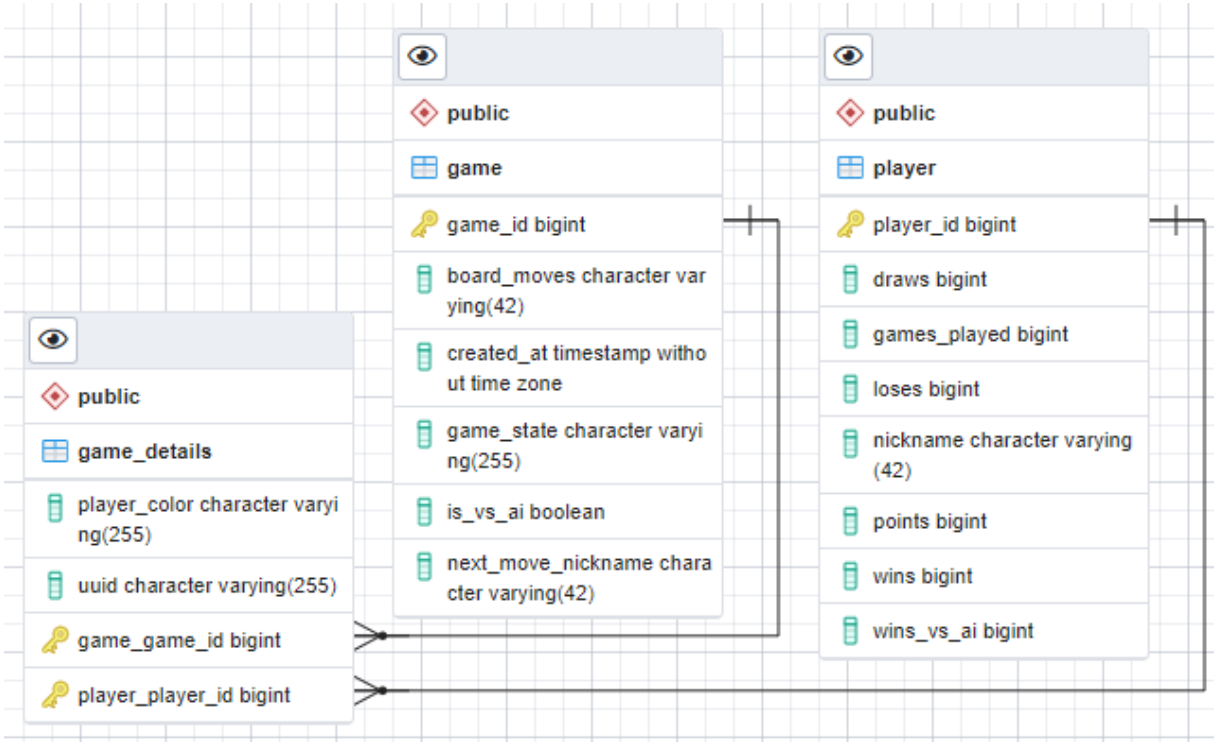
Κεφάλαιο 2ο: Αρχιτεκτονική της εφαρμογής

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει μία αναλυτική περιγραφή της αρχιτεκτονικής της εφαρμογής. Συγκεκριμένα θα αναλύσουμε αρχικά την αρχιτεκτονική της βάσης δεδομένων και τους πίνακες που δημιουργήθηκαν για να υποστηρίξουν τις ανάγκες της εφαρμογής καθώς και τις σχέσεις μεταξύ τους.

Μετέπειτα θα περιγράψουμε την δομή που ακολουθούν οι κλάσεις και οι υπηρεσίες της εφαρμογής και τις συσχετίσεις/συνδέσεις που έχουν μεταξύ τους. Τέλος θα γίνει αναφορά στην δομή του REST API της εφαρμογής καθώς και της διεπαφής που είναι ορατή στους χρήστες. Για τις προαναφερόμενες ενότητες παρατίθενται και ενδεικτικά διαγράμματα για την καλύτερη κατανόηση της αρχιτεκτονικής.

2.2 Η δομή της βάσης δεδομένων



Σχήμα 2.1: Entity Relationship Diagram

Όπως φαίνεται και στο παραπάνω στιγμιότυπο οθόνης οι πίνακες της βάσης είναι τρεις, ο πίνακας game, ο πίνακας player και ο πίνακας game_details, ο οποίος υλοποιεί την σχέση πολλά προς πολλά μεταξύ των άλλων δύο πινάκων προσθέτοντας κάποιες ακόμα στήλες. Τα πεδία του κάθε πίνακα θα αναλυθούν εκτενώς στα επόμενα υποκεφάλαια για να γίνει πλήρως κατανοητή και η χρήση τους.

2.2.1 Ο πίνακας game

Ο πίνακας game αποτελείται από τις ακόλουθες στήλες:

- game_id - είναι το id και το κύριο κλειδί του πίνακα.

- board_moves - είναι τύπου string με μέγιστη χωρητικότητα 42 χαρακτήρες. Το συγκεκριμένο string κρατάει τον αριθμό της στήλης στην οποία επέλεξε να παίξει το πούλι του ο παίκτης. Σε κάθε κίνηση προστίθεται στο τέλος του string ο αριθμός της στήλης της τελευταίας κίνησης. Έχοντας γνώση του χρώματος του κάθε παίκτη και του ότι αναγκαστικά οι παίκτες παίζουν εναλλάξ μπορούμε εύκολα να εξάγουμε όποια πληροφορία χρειάζεται σχετικά με τις κινήσεις του παιχνιδιού.

Η επιλογή να αποθηκευτούν οι κινήσεις σε ένα string έγινε μετά από σκέψη και συνυπολογίζοντας τα πλεονεκτήματα και τα μειονεκτήματα που μπορεί να έχει. Συγκεκριμένα τα κύρια πλεονεκτήματα που επηρέασαν την απόφαση ήταν τα εξής:

  - Πιο εύκολη υλοποίηση του σχεδιασμού της βάσης συγκριτικά με την υλοποίηση ενός πίνακα 2 διαστάσεων.

  - Άμεση υλοποίηση του περιορισμού σχετικά με τον μέγιστο αριθμό πουλιών που μπορεί να δεχτεί το ταμπλό (42 στην συγκεκριμένη περίπτωση).

  - Πιο φιλικό προς τον χρήστη. Δεν χρειάζεται να επιλέξει να παίξει σε θέση που ορίζεται από 2 διαστάσεις, από την στιγμή που έτσι και αλλιώς το ταμπλό έχει "βαρύτητα" και το πούλι του κάθε παίκτη αναγκαστικά θα τοποθετηθεί στην πιο κάτω ελεύθερη θέση που υπάρχει. Σε περίπτωση διαφορετικής υλοποίησης πολύ πιθανόν να έπρεπε να γίνει επιπλέον έλεγχος για την διασφάλιση της προαναφερόμενης συνθήκης.

  - Εύκολος έλεγχος ορθότητας επιλογής στήλης. Συγκεκριμένα αρκεί να ελέγχουμε ότι ο αριθμός που προστίθεται κάθε φορά στο τέλος του string είναι μεταξύ του 0 και του 6.

- created_at - στην συγκεκριμένη στήλη αποθηκεύονται το timestamp της δημιουργίας του παιχνιδιού.

- game_state - στην συγκεκριμένη στήλη αποθηκεύεται η κατάσταση του παιχνιδιού, η οποία μέσω του κώδικα java μπορεί να πάρει μόνο 3 "τιμές". Started, Running και Finished.

- is_vs_ai - η συγκεκριμένη στήλη είναι τύπου boolean και χρησιμοποιείται για να αποθηκεύεται αν το παιχνίδι παίζεται εναντίον του "έξυπνου" παίκτη.

- next_move_nickname - στην συγκεκριμένη στήλη αποθηκεύεται το όνομα του παίκτη που είναι σειρά του να παίξει.

2.2.2 Ο πίνακας player

Ο πίνακας player αποτελείται από τις ακόλουθες στήλες:

- player_id - είναι το id και το κύριο κλειδί του πίνακα.
- draws - στην συγκεκριμένη στήλη αποθηκεύεται ο αριθμός των ισοπαλιών που έχει ο παίκτης.
- games_played - στην συγκεκριμένη στήλη αποθηκεύεται ο αριθμός των παιχνιδιών που έχει παίξει ο παίκτης.
- loses - στην συγκεκριμένη στήλη αποθηκεύεται των ηττών που έχει ο παίκτης.
- nickname - στην συγκεκριμένη στήλη αποθηκεύεται το ψευδώνυμο του παίκτη το οποίο έχει τον περιορισμό να είναι μοναδικό και να μην μπορεί να είναι null.
- points - στην συγκεκριμένη στήλη αποθηκεύονται οι πόντοι του παίκτη, οι οποίοι και αυξάνονται, μειώνονται ή μένουν ίδιοι ανάλογα με την έκβαση του παιχνιδιού.
- wins - στην συγκεκριμένη στήλη αποθηκεύεται ο αριθμός των νικών που έχει ο παίκτης.
- wins_vs_ai - στην συγκεκριμένη στήλη αποθηκεύεται ο αριθμός των νικών εναντίον του "έξυπνου" παίκτη που έχει ο παίκτης.

2.2.3 Ο πίνακας game_details

Ο πίνακας game_details υλοποιεί την σχέση πολλά προς πολλά μεταξύ των δύο προηγούμενων πινάκων και έχει ως σύνθετο κύριο κλειδί τον συνδυασμό των id αυτών.
Έκτος του id, οι στήλες που υπάρχουν στον συγκεκριμένο πίνακα είναι οι ακόλουθες:

- player_color - στην συγκεκριμένη στήλη αποθηκεύεται το χρώμα που έχουν τα πούλια του παίκτη στο συγκεκριμένο παιχνίδι, το οποίο μέσω του κώδικα java μπορεί να πάρει μόνο 3 "τιμές", RED και YELLOW.
- uuid - στην συγκεκριμένη στήλη αποθηκεύεται ένας μοναδικός κωδικός ο οποίος χρησιμοποιείται για να ταυτοποιεί την ταυτότητα του κάθε παίκτη. Αυτός ο κωδικός είναι διαφορετικός για τον ίδιο παίκτη σε περίπτωση που παίζει πολλές παρτίδες ταυτοχρόνως. Συγκεκριμένα ο κωδικός ανατίθεται στον παίκτη κατά την έναρξη ή την εγγραφή σε ένα παιχνίδι.

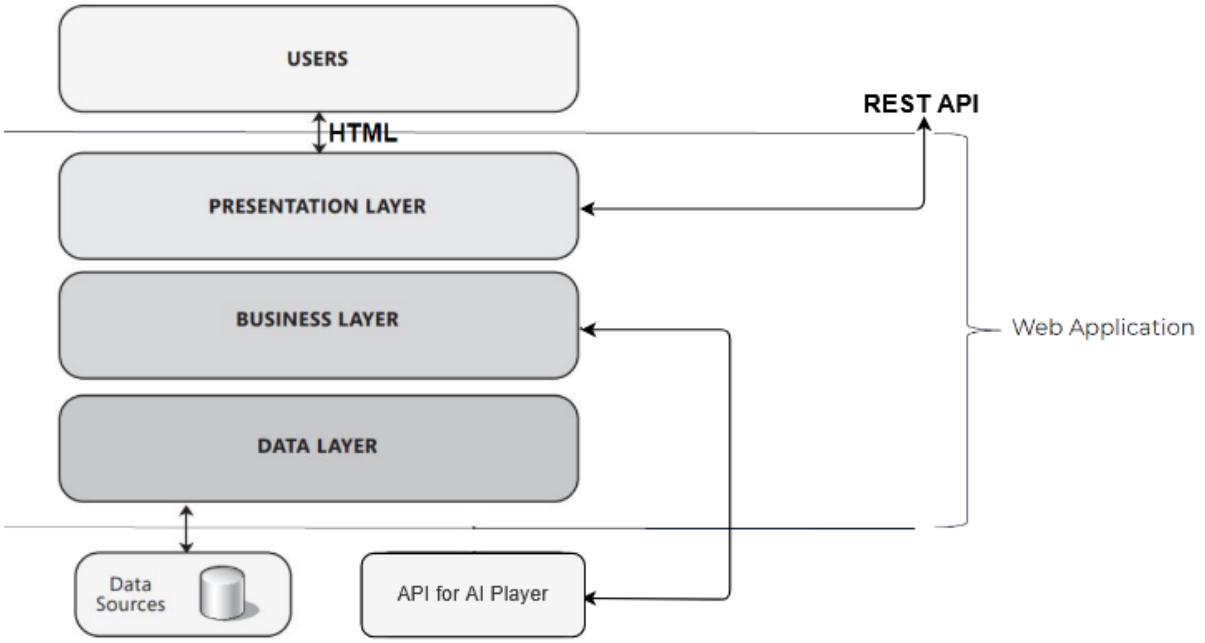
2.3 Αρχιτεκτονική του προγράμματος

2.3.1 Εισαγωγή

Στο κεφάλαιο αυτό αρχικά θα γίνει μια αναφορά στην αρχιτεκτονική των τριών επιπέδων, την αρχιτεκτονική πάνω στην οποία δομήθηκε και η εφαρμογή που ανέπτυξα.
Στην συνέχεια θα παρουσιαστεί η δομή τόσο του back-end κομματιού της εφαρμογής, όσο και του front-end και θα γίνει μια περιγραφή των περιεχομένων του κάθε module.

2.3.2 Αρχιτεκτονική τριών επιπέδων

2.3.2.1 Εισαγωγή



Σχήμα 2.2: Διάγραμμα της αρχιτεκτονικής του προγράμματος

Η βασική αρχιτεκτονική πάνω στην οποία δομήθηκε το πρόγραμμα είναι γνωστή ως αρχιτεκτονική τριών επιπέδων και η δομή της φαίνεται ξεκάθαρα και στην παραπάνω εικόνα.

Η συγκεκριμένη αρχιτεκτονική είναι μια αρχιτεκτονική για διαδικτυακές εφαρμογές που έχει ως στόχο να διαχωρίσει την εφαρμογή σε 3 επίπεδα, το επίπεδο που είναι υπεύθυνο για την παρουσίαση, το επίπεδο που είναι υπεύθυνο για την υλοποίηση της business λογικής της εφαρμογής και των υπηρεσιών της και τέλος το επίπεδο που είναι υπεύθυνο για την επικοινωνία με την βάση δεδομένων.

2.3.2.2 Presentation layer

Το presentation layer είναι υπεύθυνο για την παρουσίαση των δεδομένων στον χρήστη. Περιλαμβάνει ως δομικά του στοιχεία τους Controllers και το view της εφαρμογής. Οι Controllers είναι οι κλάσεις που έχουν ως αρμοδιότητα να ανταποκρίνονται σε συγκεκριμένα endpoints και να καλούν την αντίστοιχη μέθοδο από το business layer για να τους μεταφέρει τα απαραίτητα δεδομένα για να επιστρέψουν μια απάντηση.

Ο τρόπος που θα εμφανιστεί η απάντηση στον χρήστη είναι το view της εφαρμογής και αυτό μπορεί να είναι είτε μια HTML/CSS/JS σελίδα είτε ένα JSON αντικείμενο που θα επιστρέφεται από το REST API της εφαρμογής. Συγκεκριμένα στην εφαρμογή που ανέπτυξα υπάρχουν και 2 "δυνατότητες" που αντιστοιχίζονται σε διαφορετικά endpoints.

Καλό θα ήταν να τονιστεί ότι οι κλάσεις των Controller δεν πρέπει να έχουν καθόλου business logic στις μεθόδους τους για να είναι πλήρως διαχωρισμένες οι αρμοδιότητες που έχει κάθε επίπεδο, μιας και αυτό είναι ένα από τα βασικά πλεονεκτήματα.

2.3.2.3 Business layer

Το business layer ή αλλιώς application layer είναι υπεύθυνο για την υλοποίηση όλης της λογικής της εφαρμογής μας. Στο επίπεδο αυτό συγκαταλέγονται οι υπηρεσίες/services, δηλαδή όλες εκείνες οι κλάσεις που επεξεργάζονται τα δεδομένα βάση των επιθυμητών λειτουργιών που θέλουμε να εκτελεί η εφαρμογή.

Παραδείγματος χάρη στην εφαρμογή που ανέπτυξα, το ΣΚΟΡ 4, το business μέρος του έχει να κάνει με την διαχείριση όλων των λεπτομερειών σχετικά με την δημιουργία νέου παιχνιδιού, την εγγραφή του παίκτη, την ομαλή λειτουργία του παιχνιδιού, την διασφάλιση των κανόνων του, τον έλεγχο του νικητή, την παρουσίαση στατιστικών για κάθε παίκτη, καθώς και μια σειρά από άλλες λειτουργίες που υλοποίησα.

Πέραν αυτών των λειτουργιών, στο συγκεκριμένο επίπεδο περιλαμβάνονται και ενέργειες που μπορεί να μην σχετίζονται με ενέργειες του χρήστη, όπως μια επαναλαμβανόμενη διεργασία διαγραφής "άχρηστων" καταχωρήσεων από την βάση δεδομένων ή η αλληλεπίδραση με κάποιο εξωτερικό API με σκοπό να αποκτήσουμε χρήσιμες πληροφορίες για την εφαρμογή μας.

2.3.2.4 Data layer

Το data layer είναι υπεύθυνο για την διαχείριση και την επικοινωνία με την βάση δεδομένων. Εμπεριέχει τις κλάσεις οντοτήτων/entities καθώς και τα repositories. Οι κλάσεις entities αναπαριστούν έναν πίνακα μιας βάσης δεδομένων με σκοπό να αντιστοιχίζονται τα δεδομένα της βάσης σε ένα data access object, ένα αντικείμενο που θα φέρει την απαραίτητη πληροφορία και θα είναι εύκολα διαχειρίσιμο μέσω του

κώδικα μας. Τα repositories είναι υπεύθυνα για την υλοποίηση των μεθόδων που θα μεταφράζονται σε αιτήματα/queries προς την βάση δεδομένων.

2.3.2.5 Πλεονεκτήματα

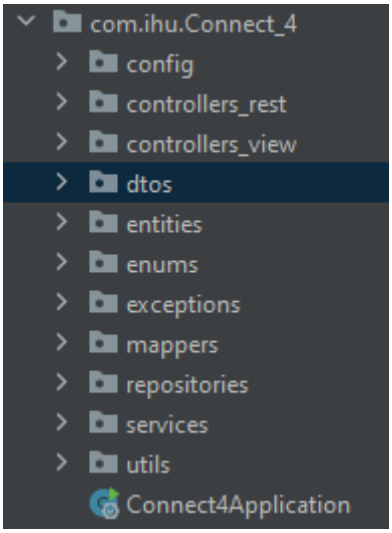
Τα βασικά πλεονεκτήματα που προσφέρει η συγκεκριμένη αρχιτεκτονική είναι τα εξής:

- Προσφέρει την δυνατότητα να αλλάζεις/αναβαθμίζεις τις τεχνολογίες που χρησιμοποιούνται σε κάθε επίπεδο, με τις μικρότερες δυνατές αλλαγές στα υπόλοιπα επίπεδα.

- Διευκολύνει την παράλληλη ανάπτυξη κώδικα από την στιγμή που το κάθε επίπεδο έχει διαφορετικές αρμοδιότητες και τα σημεία που αλληλοεξαρτώνται πρέπει να είναι το λιγότερα δυνατά.

- Διευκολύνει την συντήρηση και την ανάπτυξη νέων λειτουργιών στον κώδικα καθώς είναι πολύ πιο ξεκάθαρο που βρίσκεται το κάθε σημείο που θέλουμε να αλλάξουμε.

2.3.3 Η Δομή του back-end κομματιού της εφαρμογής



Σχήμα 2.3: Η δομή του back-end μέρους της εφαρμογής

Στο στιγμιότυπο οθόνης που υπάρχει ακριβώς πάνω φαίνονται οι φάκελοι που περιέχουν τις κλάσεις του back-end μέρους της εφαρμογής (για λόγους μεγέθους δεν φαίνονται οι κλάσεις). Θα αναφερθεί περιληπτικά ο ρόλος του κάθε φακέλου και των κλάσεων που περιέχει με βάση το επίπεδο που ανήκει στην αρχιτεκτονική τριών επιπέδων.

2.3.3.3 Entities

Ο φάκελος entities ανήκει στο data layer και περιέχει τις κλάσεις που είναι υπεύθυνες για την αναπαράσταση ενός πίνακα μιας σχεσιακής βάσης δεδομένων με σκοπό τα δεδομένα να μπορούν να μετατραπούν σε αντικείμενα αυτής της κλάσης με την βοήθεια ενός object - relational mapper. Στην περίπτωση μας, οι κλάσεις που περιέχει είναι οι εξής:

- Game
- Player
- GameDetails
- GameDetailsId (για να ορίσει το σύνθετο κύριο κλειδί του πίνακα GameDetails).

2.3.3.4 Repositories

Ο φάκελος repositories ανήκει στο data layer και περιέχει τις κλάσεις που είναι υπεύθυνες για την υλοποίηση των queries προς την βάση, είτε γράφοντας μεθόδους ακολουθώντας ένα συγκεκριμένο naming convention που ορίζει το framework είτε γράφοντας ένα sql query που το αναθέτουμε σε μια java μέθοδο. Στην περίπτωση μας, οι κλάσεις που περιέχει είναι οι εξής:

- GameRepository
- PlayerRepository
- GameDetailsRepository

2.3.3.5 Services

Ο φάκελος services ανήκει στο business layer και περιέχει τις κλάσεις που περιέχουν την κύρια υλοποίηση για τα endpoints που παρέχονται στον χρήστη μέσω των controller, τις κλάσεις που περιέχουν προγραμματισμένες διεργασίες και τις κλάσεις που υλοποιούν την επικοινωνία με το εξωτερικό API. Ο φάκελος services περιέχει τις εξής κλάσεις καθώς και τα interfaces αυτών που ορίζουν τις δημόσιες μεθόδους τους και έχουν το ίδιο όνομα χωρίς το Impl (Implementation) στο τέλος.

- PlayerServiceImpl
- GameServiceImpl
- CheatServiceImpl
- GameVsAiServiceImpl
- ScheduledTasks

2.3.3.6 Controllers

Οι φάκελοι controllers_rest και controllers_view καθώς και όλοι οι φάκελοι του επόμενου υποκεφαλαίου ανήκουν στο presentation layer.

Ο φάκελοι των controller περιέχουν τις κλάσεις που είναι υπεύθυνες για την “σύνδεση” των επιλογών του χρήστη με τις αντίστοιχες υπηρεσίες. Συγκεκριμένα, ορίζουν σε ποια endpoint θα ανταποκρίνονται, καλούν τις αντίστοιχες υπηρεσίες και επιστρέφουν την απάντηση στον χρήστη. Η διαφορά του controllers_rest και του controllers_view φακέλου, είναι ότι οι κλάσεις του πρώτου επιστρέφουν την απάντηση σε JSON μορφή ενώ του δεύτερου σε html μορφή με την βοήθεια του apache freemarker.
Ο φάκελος controllers_rest περιέχει τις κλάσεις:

- GameController
- PlayerController

Ο φάκελος controllers_view περιέχει τις κλάσεις:

- GameViewController
- PlayerViewController

2.3.3.7 Config

Ο φάκελος config περιέχει την κλάση SecurityConfig, στην οποία βρίσκονται οι ρυθμίσεις που έχουμε σχετικά με την ασφάλεια κάποιων endpoint.

2.3.3.8 Dtos

Ο φάκελος dtos περιέχει κλάσεις που ορίζουν τα πεδία των αντικειμένων που θα δεχτούμε σε ένα αίτημα ή που θα επιστρέψουμε σε μια απάντηση.

Πιο συγκεκριμένα τα DTO (Data Transfer Object) ανήκουν σε ένα “νοητό” επίπεδο μεταξύ του presentation layer και του business layer και χρησιμοποιείται τόσο για λόγους ασφαλείας, όσο και για λόγους χρηστικότητας. Στον συγκεκριμένο φάκελο υπάρχουν οι εξής κλάσεις:

- CheatDTO
- GameDTO
- PlayerDTO

2.3.3.9 Mappers

Ο φάκελος mappers περιέχει τις κλάσεις οι οποίες χρησιμοποιούνται για να μετατρέψουν ένα entity αντικείμενο σε ένα dto. Αυτή η μετατροπή γίνεται στο service layer το οποίο έχει μία εξάρτηση/dependency στους mappers.

Συγκεκριμένα στους mappers αρχικοποιούμε ένα DTO αντικείμενο και του αναθέτουμε, όσες τιμές χρειαζόμαστε με βάση τη λογική της εφαρμογής μας, από τα entity αντικείμενα. Στον συγκεκριμένο φάκελο υπάρχουν οι εξής κλάσεις:

- GameMapper
- PlayerMapper

2.3.3.10 Enums

Ο φάκελος enums περιέχει τις κλάσεις οι οποίες αναπαριστούν ομάδες από constants που χρησιμοποιούνται στην εφαρμογή. Στον συγκεκριμένο φάκελο υπάρχουν οι εξής κλάσεις:

- GameState
- PlayerColor
- SortingOrder
- SortingType

Παραδείγματος χάρη στην κλάση PlayerColor υπάρχουν οι τιμές YELLOW και RED, καθώς είναι οι μόνες από τις οποίες θα πρέπει να μπορεί να διαλέξει ο χρήστης.

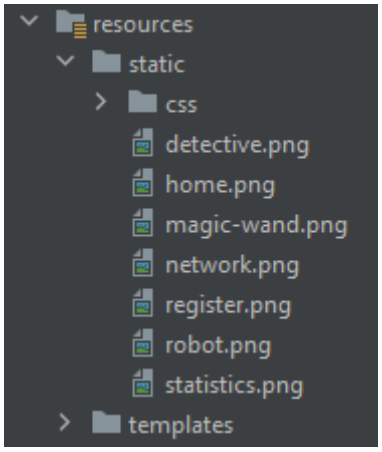
2.3.3.11 Exceptions

Ο φάκελος exceptions περιέχει τις κλάσεις οι οποίες έχουν δημιουργηθεί με σκοπό να είναι πιο ξεκάθαρη η αιτία του κάθε σφάλματος. Στον συγκεκριμένο φάκελο υπάρχουν οι εξής κλάσεις:

- FullGameException
- InvalidMoveException
- InvalidTurnPlayException
- NotAllowedNicknameException
- NotExistingGameException
- NotExistingPlayerException
- UnavailableCheatServiceException

2.3.4 Η Δομή του front-end κομματιού της εφαρμογής

2.3.4.1 Εισαγωγή



Σχήμα 2.4: Η δομή του front-end μέρους της εφαρμογής

Το front-end κομμάτι της εφαρμογής ανήκει όλο στο presentation layer.

Στο στιγμιότυπο οθόνης που υπάρχει ακριβώς πάνω φαίνονται οι φάκελοι που περιέχουν τις κλάσεις του front-end μέρους της εφαρμογής. Θα αναφερθεί περιληπτικά ο ρόλος του κάθε φακέλου και των κλάσεων που περιέχει.

2.3.4.2 Static

Στον φάκελο static υπάρχουν οι φωτογραφίες που χρησιμοποιούνται στο navbar της εφαρμογής καθώς και ο φάκελος css.

2.3.4.2.1 CSS

Ο φάκελος css περιέχει τις κλάσεις που είναι υπεύθυνες για το styling της διεπαφής. Συγκεκριμένα περιέχει τις εξής κλάσεις:

- board.css
- index.css
- form.css
- joinPage.css
- navbar.css
- statisics.css

2.3.4.3 Templates

Στον φάκελο templates υπάρχουν τα template των σελίδων που χρησιμοποιούνται για την διεπαφή καθώς και ο φάκελος parts.
Εκτός του φακέλου parts, περιέχει και τα εξής templates:

- board.ftl
- createPage.ftl
- createPageVsAi.ftl
- error.ftl
- index.ftl
- joinPage.ftl
- registerPage.ftl
- statistics.ftl

2.3.4.4 Parts

Ο φάκελος parts περιέχει τα templates που αποτελούν μέρος σε περισσότερες από μια σελίδες. Γι' αυτό τον λόγο υπάρχουν σαν ξεχωριστά parts για να επαναχρησιμοποιούνται και να μην ξαναγράφεται ο ίδιος κώδικας. Τα templates που περιέχονται σε αυτόν τον φάκελο είναι τα εξής:

- navbar.ftl
- footer.ftl
- detectiveInfo.ftl

Κεφάλαιο 3ο: Το REST API της εφαρμογής

3.1 Εισαγωγή

Στο συγκεκριμένο κεφάλαιο θα γίνει μια εισαγωγή στο τι είναι ένα REST API και θα παρουσιαστεί το REST API που αναπτύχθηκε στα πλαίσια αυτής της εφαρμογής.
Για την εισαγωγή στο τι είναι ένα REST API θα χρειαστεί να αρχικά να αναλύσουμε κάποιες ακόμα τεχνολογίες και έννοιες με σκοπό να σχηματιστεί ολοκληρωμένη εικόνα.

3.2 Τι είναι API

Ένα API ή αλλιώς μια διεπαφή προγραμματισμού εφαρμογών είναι ένα σύνολο πρωτοκόλλων και ορισμών. Τα API επιτρέπουν σε μια εφαρμογή ή μια υπηρεσία να αλληλεπιδρά και να ανταλλάσσει δεδομένα με άλλα προγράμματα. Τα δεδομένα που ανταλλάσσονται μέσω ενός API μπορούν να είτε XML μορφή είτε JSON.

Η χρήση των API μας επιτρέπει να λαμβάνουμε και να χειριζόμαστε πληροφορίες χωρίς να μας ενδιαφέρει το πως παράγονται και υλοποιούνται. Καλώντας ένα καλά δομημένο και τεκμηριωμένο API ξέρουμε ακριβώς την δομή και την μορφή της πληροφορίας που θα μας επιστραφεί.

Τα APIs χρησιμοποιούνται πολύ συχνά γιατί μπορούν να διευκολύνουν την ανάπτυξη ενός λογισμικού σε πολλά επίπεδα. Τα πιο βασικά πλεονεκτήματά του έχουν να κάνουν με την ευελιξία και την επαναχρησιμοποίηση που παρέχουν.

3.3 HTTP

Το Πρωτόκολλο Μεταφοράς Υπερκειμένου ή αλλιώς HyperText Transfer Protocol / HTTP αποτελεί ένα πρωτόκολλο επικοινωνίας. Το πρωτόκολλο αυτό σχεδιάστηκε για να ορίσει κάποιους κανόνες σχετικά με την μεταφορά δεδομένων και την ανταλλαγή μηνυμάτων στον παγκόσμιο ιστό.

Το πρωτόκολλο αυτό πλέον αποτελεί το κυρίαρχο στον παγκόσμιο ιστό και οι browsers το έχουν ως προεπιλογή, αν δεν οριστεί κάτι διαφορετικό από τον χρήστη.

Το HTTP ορίζει μεθόδους ή αλλιώς ρήματα για να υποδείξει την επιθυμητή ενέργεια που πρέπει να εκτελεστεί στον προσδιορισμένο πόρο. Το αν αυτός ο πόρος αντιπροσωπεύει προϋπάρχοντα δεδομένα ή δεδομένα που δημιουργούνται δυναμικά, εξαρτάται από την υλοποίηση του διακομιστή.

Οι βασικές μέθοδοι που ορίζονται από το HTTP είναι οι εξής:

- GET - Η μέθοδος GET ζητά μια αναπαράσταση του καθορισμένου πόρου. Τα αιτήματα που χρησιμοποιούν το GET θα πρέπει να ανακτούν μόνο δεδομένα και να μην έχουν άλλο αποτέλεσμα.

- HEAD - Η μέθοδος HEAD ζητά μια απόκριση ίδια με εκείνη ενός αιτήματος GET, αλλά χωρίς το σώμα απόκρισης. Αυτό είναι χρήσιμο για την ανάκτηση μετα-πληροφοριών γραμμένων σε κεφαλίδες απόκρισης, χωρίς να χρειάζεται να μεταφέρετε ολόκληρο το περιεχόμενο.

- POST - Η μέθοδος POST ζητά από τον διακομιστή να αποδεχτεί την οντότητα που περιλαμβάνεται στο αίτημα ως νέο δευτερεύον στοιχείο του πόρου του ιστού που προσδιορίζεται από το URL, και να τα μεταφέρει στον Server, όχι απλά να γίνει ανάκτηση. Δηλαδή η μέθοδος POST κυρίως αφορά ενέργειες σχετικές με την αποθήκευση νέας πληροφορίας στον server ή στην βάση δεδομένων.

- PUT - Η μέθοδος PUT τροποποιεί τα δεδομένα που υπάρχουν στην σελίδα αν υπάρχουν ήδη αλλιώς δημιουργεί αυτόν τον νέο πόρο. Με άλλα λόγια, ζητά την αποθήκευση της κλειστής οντότητας κάτω από το παρεχόμενο URI. Εάν το URI αναφέρεται σε έναν ήδη υπάρχοντα πόρο, τροποποιείται. Εάν το URI δεν δείχνει έναν υπάρχοντα πόρο, τότε ο server μπορεί να τον δημιουργήσει.

- DELETE - Η μέθοδος DELETE κάνει αυτό ακριβώς που περιγράφει, δηλαδή διαγράφει τον συγκεκριμένο πόρο που έχει ζητήσει ο πελάτης, αν βέβαια υπάρχει.

3.4 Αρχιτεκτονική Client Server

Η αρχιτεκτονική Client Server είναι μία αρχιτεκτονική, στην οποία ο φόρτος εργασίας και οι διεργασίες κατανέμονται ανάμεσα στους λεγόμενους Servers ή αλλιώς πάροχοι Υπηρεσιών/Εξυπηρετητές και στους Clients ή αλλιώς πελάτες που κάνουν την αίτηση για κάποια υπηρεσία. Συγκεκριμένα το σύστημα Client Server αποτελείται από αυτές τις δύο πλευρές οι οποίες αλληλεπιδρούν μεταξύ τους στην λογική των Request-Response, δηλαδή της αίτησης για μία ενέργεια και της ανταπόκρισης δηλαδή της απάντησης προς την αίτηση αυτή.

3.4.1 Client

Ο Client δηλαδή ο πελάτης είναι αυτός που κάνει την αίτηση και μπορεί να είναι ένας υπολογιστής, ένα κινητό, ένα tablet ή ο browser ενός υπολογιστή. Τα βασικά στοιχεία του έχουν να κάνουν με την δυνατότητα του να εμφανίζει την πληροφορία με τρόπο φιλικό προς το χρήστη "τρέχοντας" το λογισμικό που είναι απαραίτητο για την γραφική διεπαφή, να μπορεί να υλοποιεί τις αιτήσεις προς στον server και να μπορεί να αποθηκεύει κάποιες χρήσιμες πληροφορίες εφόσον αυτές επιστραφούν.

3.4.2 Server

Η κύρια λειτουργία η οποία πρέπει να μπορεί να υλοποιεί ο Server είναι η απάντηση των αιτήσεων του πελάτη. Για να το καταφέρει θα πρέπει να μπορεί να υλοποιεί μια σειρά από άλλες λειτουργίες όπως οι ακόλουθες, να μπορεί να αποθηκεύει, να προστατεύει και να ανακτά τις πληροφορίες που δέχεται, όπως επίσης και να τις επεξεργάζεται.

Για την επικοινωνία μεταξύ client και server πρέπει να υπάρχουν σαφώς ορισμένοι κανόνες σχετικά με την δομή και την μορφή των μηνυμάτων που ανταλλάσσονται. Αυτοί λοιπόν οι κανόνες και η γλώσσα ορίζονται σε ένα πρωτόκολλο επικοινωνιών.

Τα πρωτόκολλα που αφορούν το μοντέλο client και Server ορίζονται στο επίπεδο εφαρμογής ή αλλιώς Application Layer. Στο επίπεδο εφαρμογής ορίζονται τα βασικά πρότυπα διαλόγου.

Για την περαιτέρω διευκόλυνση της επικοινωνίας αυτής, ο Server ορίζει και ένα API το οποίο είναι ένα επίπεδο αφαίρεσης για την πρόσβαση σε μία υπηρεσία. Το API έτσι διευκολύνει την ανταλλαγή πληροφορίας μεταξύ των διαφόρων πλατφορμών και αποτελεί ένα πολύ χρήσιμο εργαλείο σε τέτοιου είδους διαδικασίες.

3.5 Τι είναι REST API

"Η αρχιτεκτονική REST (**RE**presentational **S**tate **T**ransfer) αποτελεί ένα σύνολο από αρχές σχεδίασης μιας δικτυακής υπηρεσίας. Η μεταβολή της κατάστασης της βάσης δεδομένων του συστήματος περιγράφεται και μεταφέρεται στο σύστημα μέσω του πρωτοκόλλου **HTTP** από διάφορους clients ανεξαρτήτως της γλώσσας στην οποία έχουν υλοποιηθεί."
Οι βασικοί κανόνες του REST είναι οι εξής:

- Διαχωρισμός της εφαρμογής του client και της υλοποίησης του server (client - server architecture).
- Η μεταφορά δεδομένων πρέπει να γίνεται μέσω του HTTP.
- Πρέπει να είναι stateless, δηλαδή οποιαδήποτε αίτηση σε μια REST υπηρεσία δε πρέπει να εξαρτάται ή να σχετίζεται με οποιαδήποτε άλλη αίτηση.

3.6 Το REST API της συγκεκριμένης εφαρμογής

Στα πλαίσια της ανάπτυξης της συγκεκριμένης εφαρμογής, υλοποιήθηκε και ένα REST API που παρέχει όλες τις λειτουργίες που υπάρχουν στην διεπαφή της εφαρμογής και μπορεί να χρησιμοποιηθεί από χρήστες ή εφαρμογές.

Παρακάτω παρατίθενται οι πίνακες με τα διαθέσιμα REST endpoints καθώς και μια περιγραφή των λειτουργιών, των επιστρεφόμενων πόρων και των πιθανών HTTP κωδικών που μπορεί να επιστραφούν.

3.6.1 Το API για τους παίκτες

ENDPOINT	METHOD	PARAM TYPE	DESCRIPTION	RETURNS	STATUS
/api/players/{nickname}	POST	nickname-String	Register the player if does not exist	PlayerDTO	200(OK)
/api/players/statistics	GET	Request param: sortingType optional, enum (points, wins,loses, draws, gamesPlayed, winsVsAi). Request param: sortingOrder optional, enum (asc, desc)	Returns sorted statistics for all the players	List of PlayerDTO	200(OK)

3.6.2 Η δομή του PlayerDTO

```
"PlayerDTO": {
        "type": "object",
        "properties": {
          "draws": {
            "type": "integer"
          },
          "gamesPlayed": {
            "type": "integer"
          },
          "loses": {
            "type": "integer"
          },
          "nickname": {
            "type": "string"
          },
          "points": {
            "type": "integer"
          },
```

```
        "wins": {
          "type": "integer"
        },
        "winsVsAi": {
          "type": "integer"
        }
      },
      "title": "PlayerDTO"
    }
```

3.6.3 Το API για το παιχνίδι

ENDPOINT	METHOD	PARAM TYPE	DESCRIPTION	RETURNS	STATUS
/api/games/{nickname}	POST	nickname-String	Create a game for the user with the specific nickname	GameResponseDTO	200(OK), 400(BAD REQUEST)
/api/games	GET		Rerurns all available games to join	List of GameResponseDTO	200(OK)
/api/games/join/{nickname}/{id}	POST	nickname-String, id-Intrger	Add player with the specific nickname to the game with the specific id	GameResponseDTO	200(OK), 400(BAD REQUEST), 401(UNAUTHORIZED)
/api/games/play/{nickname}/{uuid}/{id}/{column}	POST	nickname-String, uuid-String, id-Integer, column-Integer	Plays for the specific player with this uuid to the game with this id the specific move	GameResponseDTO	200(OK), 400(BAD REQUEST), 401(UNAUTHORIZED)
/api/games/{nickname}/{uuid}/{id}	GET	nickname-String, id-Integer	Returns all the needed informati	GameResponseDTO	200(OK), 401(UNAUTHORIZED)

			on about the game		
/api/games/cheat/{nickname}/{uuid}/{id}	POST	nickname-String, uuid-String, id-Integer	Plays the best move for the player	GameResponseDTO	200(OK), 400(BAD REQUEST), 401(UNAUTHORIZED)
/api/games/ai/{nickname}	POST	nickname-String	Create a game for the user with the specific nickname versus AI-Robot	GameResponseDTO	200(OK), 400(BAD REQUEST)
/api/games/ai/play/{nickname}/{uuid}/{id}/{column}	POST	nickname-String, uuid-String, id-Integer, column-Integer	Plays for the specific player with this uuid to the game with this id the specific move and opponent the AI-Robot	GameResponseDTO	200(OK), 400(BAD REQUEST), 401(UNAUTHORIZED)
/api/games/ai/cheat/{nickname}/{uuid}/{id}	POST	nickname-String, uuid-String, id-Integer	Plays the best move for the player in a game versus AI-Robot	GameResponseDTO	200(OK), 400(BAD REQUEST), 401(UNAUTHORIZED)

/api/games/needsUpdate/{id}/{numOfMoves}	GET	id-Intrger, numOfMoves-Integer	True if the game with this id and this numOfmoves is not up to date, False otherwise	Boolean	200(OK)

3.6.4 Η δομή του GameDTO

```
"GameResponseDTO": {
        "type": "object",
        "properties": {
          "board": {
            "type": "array",
            "items": {
              "type": "array",
              "items": {
                "type": "string"
              }
            }
          },
          "gameState": {
            "type": "string"
          },
          "id": {
            "type": "integer"
          },
          "nextMoveNickname": {
            "type": "string"
          },
          "redPlayerNickname": {
            "type": "string"
          },
          "uuid": {
            "type": "string"
          },
          "yellowPlayerNickname": {
```

```
            "type": "string"
         },
         "isVsAi": {
            "type": "boolean"
         }
      },
      "title": "GameResponseDTO"
   }
```

Κεφάλαιο 4ο: Εγχειρίδιο εφαρμογής

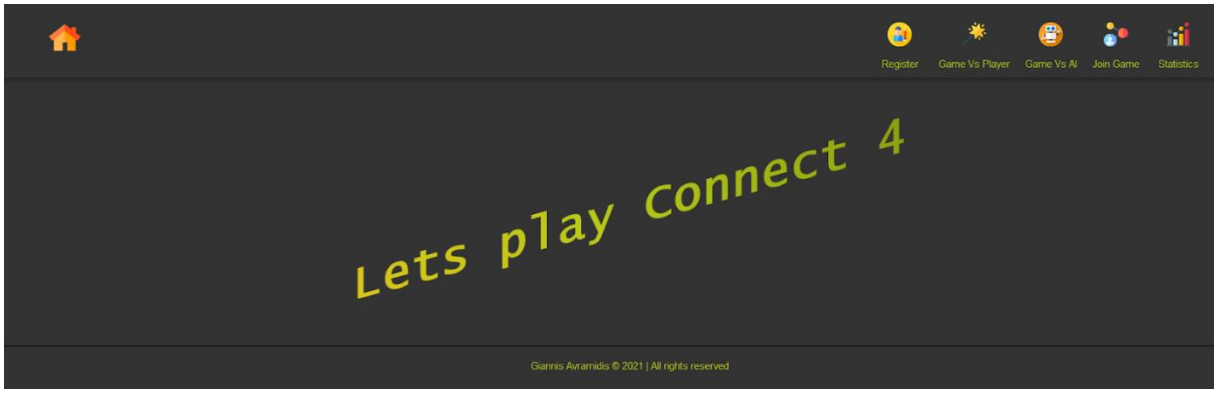
4.1 Εισαγωγή

Στο παρόν κεφάλαιο θα περιγράφουν εκτενώς όλες οι λειτουργίες της εφαρμογής και οι δυνατότητες που έχει ο χρήστης. Θα συνοδευτούν από τα αντίστοιχα στιγμιότυπα οθόνης ώστε να σχηματιστεί ξεκάθαρη εικόνα για όλη την εφαρμογή.

Η εφαρμογή είναι διαθέσιμη στο https://app-connect-4.herokuapp.com/

Ο κώδικας της εφαρμογής καθώς και πληροφορίες για το πώς μπορεί κάποιος να το τρέξει τοπικά στον υπολογιστή του βρίσκεται στο https://github.com/giannisav/connect-4

4.2 Αρχική σελίδα



Σχήμα 7: Η αρχική σελίδα της εφαρμογής

Στο πάνω μέρος της αρχικής σελίδας υπάρχει το navigation bar. Το navigation bar αποτελείται από τα εξής μέρη:

- Μια εικόνα που απεικονίζει ένα σπιτάκι η οποία λειτουργεί ως σύνδεσμος για την αρχική σελίδα.

- Μια εικόνα που απεικονίζει έναν άνθρωπο με ένα μολύβι η οποία λειτουργεί ως σύνδεσμος για τη σελίδα εγγραφή νέου χρήστη.

- Μια εικόνα που απεικονίζει ένα ραβδί η οποία λειτουργεί ως σύνδεσμος για τη σελίδα δημιουργίας νέου παιχνιδιού εναντίον άλλου παίκτη.

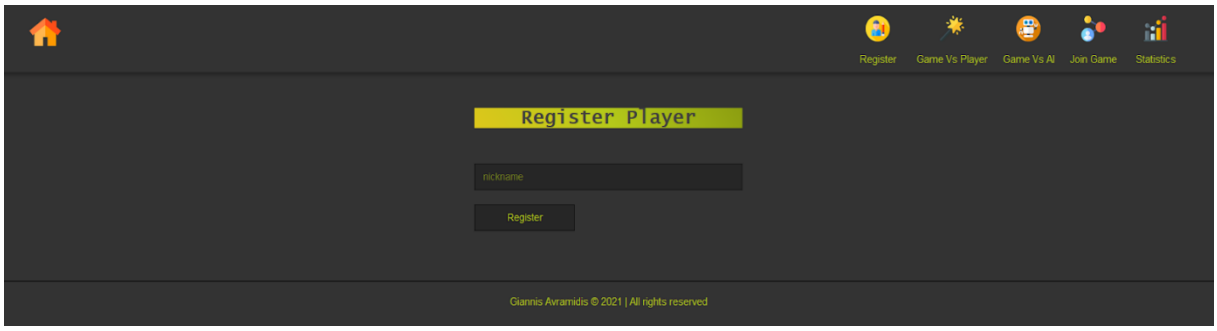
- Μια εικόνα που απεικονίζει ένα ρομπότ η οποία λειτουργεί ως σύνδεσμος για τη σελίδα δημιουργίας νέου παιχνιδιού εναντίον του "έξυπνου" παίκτη.

- Μια εικόνα που απεικονίζει τρεις συνδεδεμένους κύκλους η οποία λειτουργεί ως σύνδεσμος για τη σελίδα εύρεσης παιχνιδιών με κενές θέσεις και σύνδεσης σε αυτά.

- Μια εικόνα που απεικονίζει ένα γράφημα η οποία λειτουργεί ως σύνδεσμος για τη σελίδα απεικόνισης των στατιστικών των παικτών.

Εκτός του navigation bar, υπάρχει στο κέντρο της σελίδας με μεγάλα γράμματα η έκφραση Let's play connect 4 και στο κάτω μέρος της σελίδας υπάρχει το footer με το ονοματεπώνυμό μου και το έτος που διανύουμε.

4.3 Σελίδα εγγραφής νέου παίκτη



Σχήμα 8: Η σελίδα εγγραφή νέου παίκτη

Στη σελίδα εγγραφής νέου παίκτη υπάρχει πάλι, όπως και σε όλες τις σελίδες, το navigation bar στο πάνω μέρος της. Η σελίδα έχει ένα label που αναγράφει την φράση “Register Player”, έχει μία φόρμα ώστε ο χρήστης να εισάγει το nickname του και το κουμπί Register το οποίο αποθηκεύει τον χρήστη με το nickname που έχει δηλώσει.

Για να δημιουργηθεί ο νέος χρήστης πρέπει να περάσει από το validation που πραγματοποιείται στο back-end μέρος της εφαρμογής και ελέγχει το μέγεθος του nickname καθώς και το αν περιέχει χαρακτήρες html οι οποίοι ευθύνονται για ευπάθειες σχετικά με XSS επιθέσεις.

Στην περίπτωση που δεν είναι αποδεκτό το nickname, εμφανίζει το αντίστοιχο μήνυμα λάθους.
Στην περίπτωση που το nickname είναι αποδεκτό, αν δεν υπάρχει ίδιο nickname στη βάση δεδομένων, δημιουργείται ένας νέος παίκτης έχοντας κάποια προεπιλεγμένα πεδία. Για παράδειγμα το πεδίο "points" είναι ορισμένο στους χίλιους πόντους, το πεδίο "games played" είναι ορισμένο στο μηδέν, όπως και τα πεδία "wins", "loses", draws και "wins_vs_ai".

4.4 Σελίδα δημιουργίας παιχνιδιού εναντίον άλλου παίκτη

Σχήμα 9: Η σελίδα δημιουργίας παιχνιδιού εναντίον άλλου παίκτη

Στην σελίδα δημιουργίας παιχνιδιού εναντίον άλλου παίκτη, εκτός του navigation bar, υπάρχει μια φόρμα ίδιας αισθητικής με την προηγούμενη σελίδα μόνο που το label αναγράφει την φράση “Game Vs Player” και το κουμπί γράφει “Create”.

Σε περίπτωση που ο χρήστης εισάγει το nickname ενός ήδη εγγεγραμμένου χρήστη και πατήσει το κουμπί, θα δημιουργηθεί ένα καινούργιο παιχνίδι και ο χρήστης θα μεταφερθεί στην κύρια σελίδα του παιχνιδιού.

Σε περίπτωση που ο χρήστης εισάγει ένα nickname που δεν υπάρχει ανάμεσα στους εγγεγραμμένους χρήστες, τότε θα εμφανιστεί το αντίστοιχο μήνυμα.

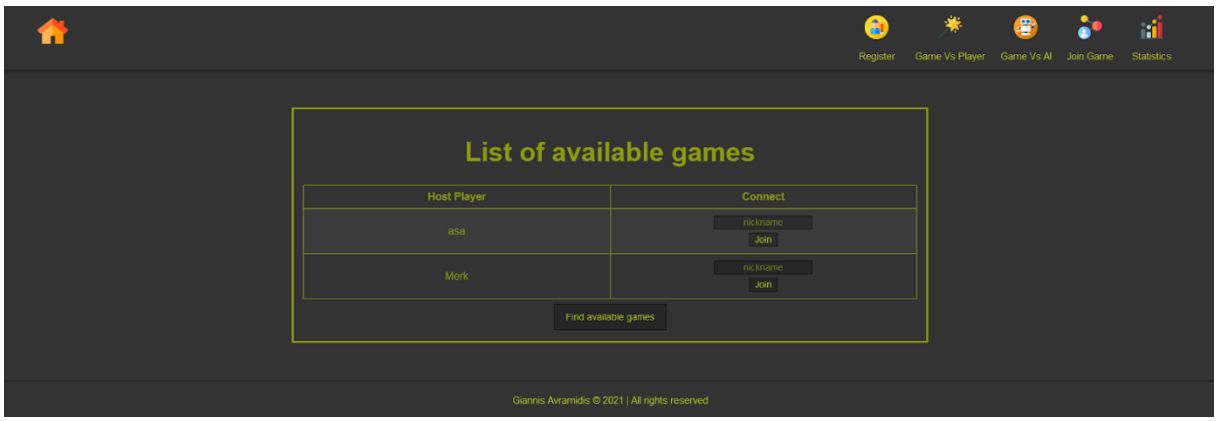
4.5 Σελίδα δημιουργίας παιχνιδιού εναντίον του “έξυπνου” παίκτη

Σχήμα 10: Η σελίδα δημιουργίας παιχνιδιού εναντίον του “έξυπνου” παίκτη

Η σελίδα δημιουργίας παιχνιδιού εναντίον του "έξυπνου" παίκτη είναι πανομοιότυπη με με την σελίδα δημιουργίας παιχνιδιού εναντίον κανονικού παίκτη με τη μικρή διαφορά ότι στο label από την φόρμα εισαγωγής nickname, αναγράφει "Game Vs AI".

Σε περίπτωση πού το nickname γίνει αποδεκτό, τότε ο χρήστης θα μεταφερθεί στην κυρία σελίδα παιχνιδιού και αυτομάτως θα έχει ως αντίπαλο έναν παίκτη με την ονομασία "AI-Robot" ο οποίος παίζει πάντα την καλύτερη δυνατή κίνηση καλώντας ένα εξωτερικό API που υλοποιεί έναν alpha-beta αλγόριθμο.

4.6 Σελίδα εύρεσης παιχνιδιών και εγγραφής σε αυτά



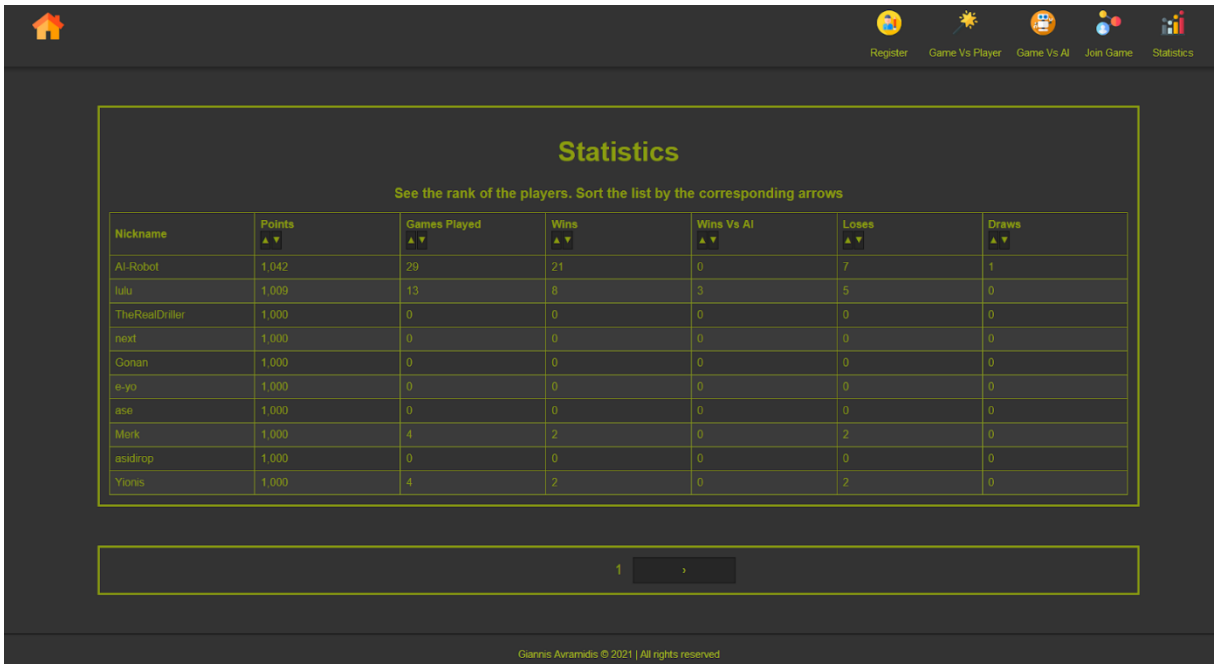
Σχήμα 11: Η σελίδα εύρεσης παιχνιδιών και εγγραφής σε αυτά

Η παραπάνω σελίδα απεικονίζει έναν πίνακα ο οποίος έχει τίτλο “List of available games” και στο κάτω μέρος έχει ένα κουμπί που αναγράφει “Find available games”. Μόλις ο χρήστης πατήσει το κουμπί εμφανίζονται οι 2 στήλες του πίνακα, η αριστερή αναγράφει “Host Player” και η δεξιά “Connect”, μαζί με όλα τα διαθέσιμα παιχνίδια εκείνη την χρονική στιγμή. Ως διαθέσιμα παιχνίδια έχουν οριστεί όσα έχουν δημιουργηθεί τα τελευταία δέκα λεπτά και έχουν μόνο έναν παίκτη μέσα στο “δωμάτιο”παιχνιδιού.

Στην δεξιά στήλη, δίπλα από το nickname του Host Player, απεικονίζεται μια φόρμα η οποία έχει ένα πεδίο εισαγωγής nickname και από κάτω ένα κουμπί “Join”.

Σε περίπτωση που το παιχνίδι είναι γεμάτο την στιγμή που ο χρήστης κάνει click στο κουμπί “Join” ή σε περίπτωση που ο χρήστης εισάγει nickname που δεν είναι εγγεγραμμένο στην βάση δεδομένων, τότε το αντίστοιχο μήνυμα λάθους θα εμφανιστεί στην οθόνη του χρήστη.
Σε περίπτωση που ολοκληρωθεί επιτυχώς η εγγραφή, ο νέος παίκτης μεταφέρεται στην κύρια σελίδα του παιχνιδιού και το παιχνίδι είναι πλέον έτοιμο για να αρχίσει.

4.7 Σελίδα εμφάνισης των στατιστικών των παικτών



Σχήμα 12: Η σελίδα με τα στατιστικά των παικτών

Η σελίδα εμφάνισης των στατιστικών των παικτών περιέχει έναν πίνακα ο οποίος έχει τον τίτλο “Statistics”, τον υπότιτλο “See the rank of the players. Sort the list by the corresponding arrows”, και τις παρακάτω στήλες:

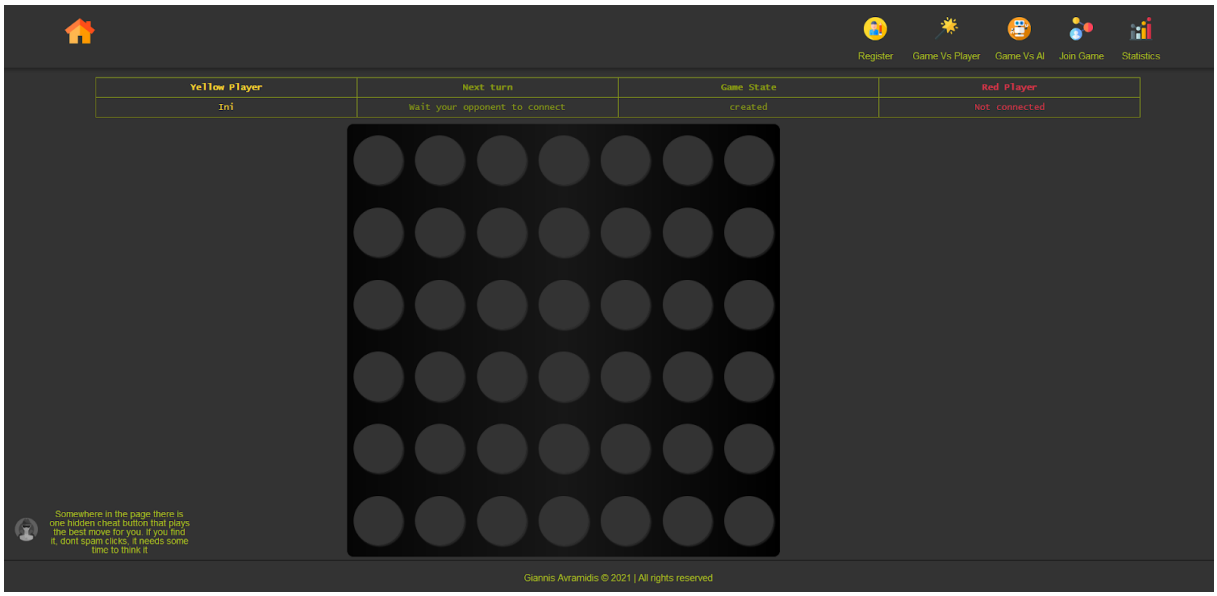
- Nickname
- Points
- Games Played
- Wins
- Wins Vs AI
- Loses
- Draws

Οι στήλες αυτές αποτελούν κάποια από τα πεδία του κάθε παίκτη και οι γραμμές του πίνακα αυτού είναι όλοι οι αποθηκευμένοι παίκτες. Κάτω από το όνομα κάθε στήλης υπάρχουν 2 βέλη, ένα που δείχνει προς τα πάνω και ένα προς τα κάτω, τα οποία χρησιμοποιούνται για να ταξινομηθούν οι παίκτες βάση του συγκεκριμένου πεδίου σε αύξουσα ή φθίνουσα σειρά αντίστοιχα.

Πέραν αυτού η συγκεκριμένη σελίδα λειτουργεί βάση σελιδοποίησης. Συγκεκριμένα, όπως φαίνεται και στην εικόνα, στο κάτω μέρος υπάρχει ο αριθμός της σελίδας και ένα βελάκι προς κάθε πλευρά στην όποια έχει δυνατότητα ο χρήστης να περιηγηθεί. Η κάθε σελίδα είναι διαμορφωμένη ώστε να δέχεται μέχρι 10 καταχωρήσεις στον πίνακα και οι υπόλοιπες καταχωρήσεις να μετατίθενται σε επόμενες σελίδες.

4.8 Κύρια σελίδα παιχνιδιού

Στην κύρια σελίδα του παιχνιδιού παρατίθενται 3 στιγμιότυπα οθόνης για να μπορέσουν να καλύψουν όλες τις λειτουργίες που υπάρχουν σε αυτήν.



Σχήμα 13: Η κύρια σελίδα του παιχνιδιού (1)

Αρχικά βλέπουμε ότι στο πάνω μέρος αναγράφονται μέσα σε έναν πίνακα, από τα αριστερά προς τα δεξιά, τα εξής στοιχεία:

- Το όνομα του κίτρινου παίκτη.

- Το όνομα του παίκτη που είναι η σειρά του να παίξει στον επόμενο γύρο (Σε περίπτωση που δεν έχει εγγραφεί ακόμα δεύτερος παίκτης αναγράφει “wait an opponent to connect” και σε περίπτωση που το παιχνίδι έχει λήξει αναφέρει είτε το όνομα του νικητή είτε ότι το αποτέλεσμα είναι ισοπαλία.

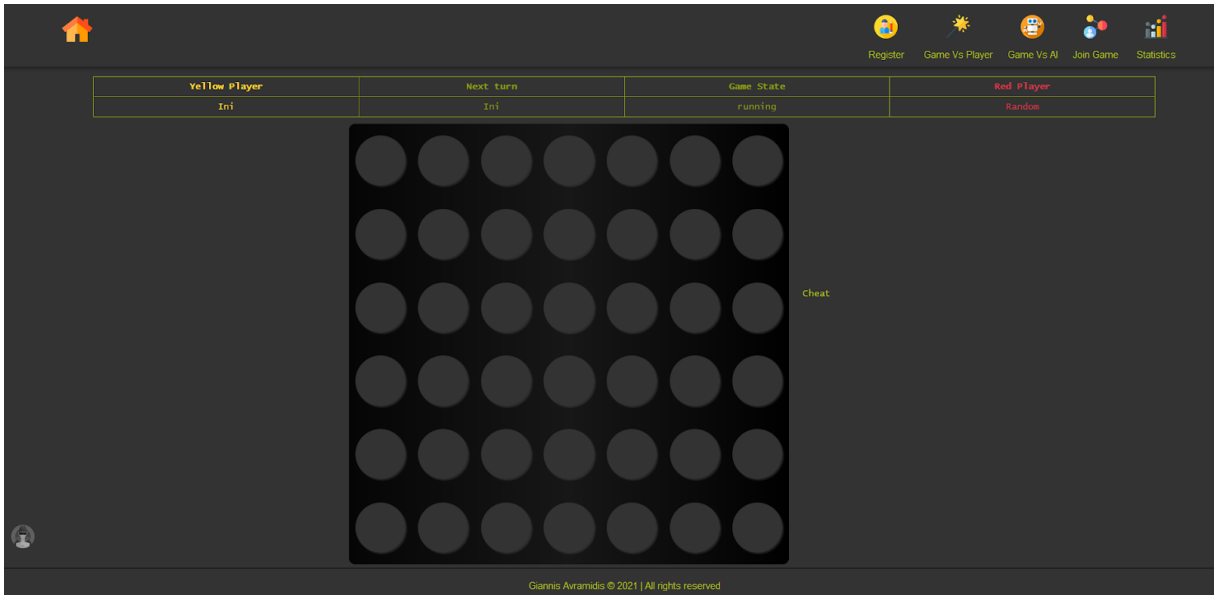
- Η κατάσταση του παιχνιδιού η οποία παίρνει 3 τιμές:

1. created - Όταν το παιχνίδι έχει δημιουργηθεί αλλά δεν έχει συνδεθεί ακόμα δεύτερος παίκτης.
2. running - Όταν το παιχνίδι “τρέχει”, δηλαδή καθ’ όλη την διάρκεια του παιχνιδιού.
3. finished - Όταν το παιχνίδι τελειώσει, ανεξαρτήτως αποτελέσματος.

- Το όνομα του κόκκινου παίκτη ή not connected σε περίπτωση που δεν έχει συνδεθεί ακόμη κάποιος παίκτης.

Στην συνέχεια στο κέντρο της οθόνης υπάρχει το ταμπλό του παιχνιδιού, με μέγεθος 6 γραμμών και 7 στηλών.

Τέλος, στο παραπάνω στιγμιότυπα οθόνης φαίνεται κάτω αριστερά το εικονίδιο ενός μυστικού πράκτορα που κρατάει μια εφημερίδα. Όταν ο χρήστης αφήσει τον δείκτη του ποντικιού πάνω του εμφανίζει το ακόλουθο μήνυμα “Somewhere in the page there is one hidden cheat button that plays the best move for you. If you find it, don't spam clicks, it needs some time to think about it”.



Σχήμα 14: Η κύρια σελίδα του παιχνιδιού (2)

Στο παραπάνω στιγμιότυπα οθόνης φαίνεται ότι έχει συνδεθεί κάποιος παίκτης και κατ’ επέκταση έχει ανανεωθεί το όνομα του παίκτη που παίζει στον επόμενο γύρο, η κατάσταση του παιχνιδιού και το όνομα του κόκκινου παίκτη. Επίσης φαίνεται καλύτερα το εικονίδιο στο κάτω αριστερό μέρος της οθόνης. Τέλος περνώντας τον δείκτη του ποντικιού πάνω από το σημείο που βρίσκεται το “κρυφό” κουμπί (δεξιά από το ταμπλό και λίγο πιο πάνω από την μέση), αυτό εμφανίζεται. Πατώντας το, αναλαμβάνει να παίξει την καλύτερη δυνατή κίνηση για τον χρήστη καλώντας ένα εξωτερικό API για το οποίο θα μιλήσουμε στο επόμενο κεφάλαιο.

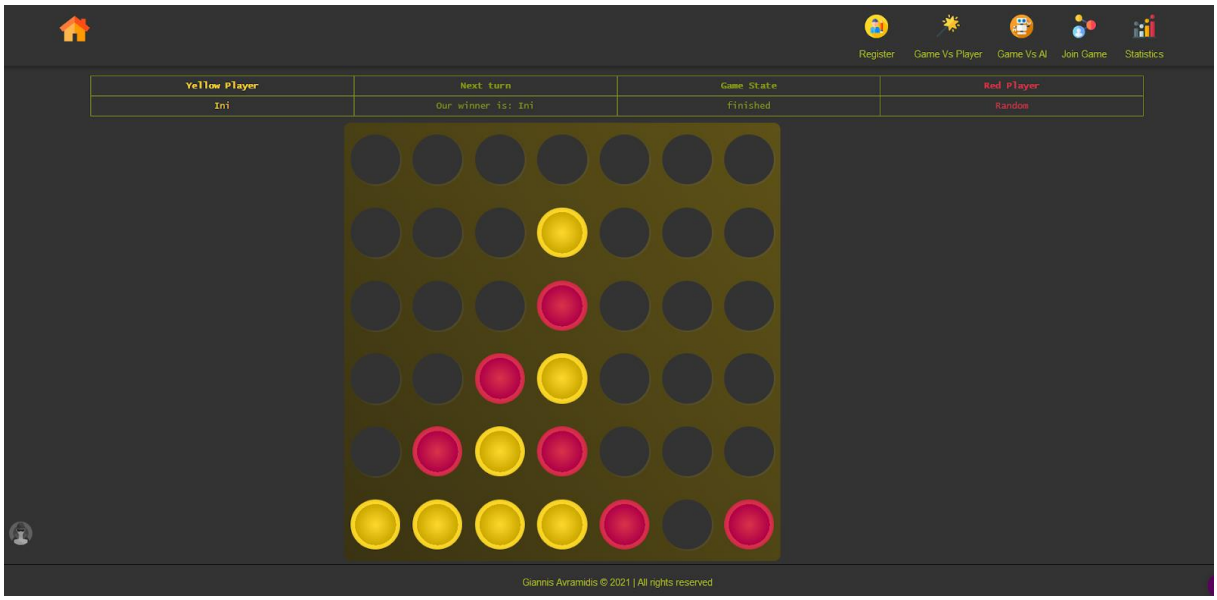
Η διαδικασία για να παίξει μια κίνηση ο παίκτης είναι απλή. Αρκεί να κάνει κλικ σε έναν οποιοδήποτε κύκλο του ταμπλό και ένα πούλι θα εμφανιστεί στην πιο χαμηλή ελεύθερη θέση εκείνης της στήλης, εφόσον υπάρχει και δεν είναι γεμάτη η στήλη. Σε περίπτωση που είναι γεμάτη δεν θα γίνει δεκτή κάποια κίνηση.

Κάθε φορά που γίνεται μια κίνηση αυτή περνάει από μια σειρά ελέγχων που διασφαλίζουν τους κανόνες του παιχνιδιού. Οι βασικοί έλεγχοι που γίνονται είναι οι εξής:

- Ελέγχεται αν το παιχνίδι είναι σε κατάσταση running.

- Ελέγχεται αν υπάρχει παίκτης με το συγκεκριμένο nickname.
- Ελέγχεται αν το nickname του παίκτη είναι αποθηκευμένο στους παίκτες του συγκεκριμένου παιχνιδιού και επιβεβαίωση της ταυτότητας του παίκτη μέσω ενός μοναδικού κωδικού που του ανατίθεται για κάθε παιχνίδι.
- Ελέγχεται αν είναι η σειρά του συγκεκριμένου παίκτη για να παίξει.
- Ελέγχεται αν ο αριθμός της στήλης που επιλέχθηκε είναι έγκυρος, καθώς υπάρχουν μόνο 7 στήλες.
- Ελέγχεται αν η στήλη που επιλέχθηκε έχει άδειες θέσεις.

Αφού γίνουν οι προαναφερόμενοι έλεγχοι, καταχωρείται η κίνηση και ελέγχεται αν η κίνηση οδηγεί σε λήξη του παιχνιδιού είτε λόγω νίκης κάποιου παίκτη είτε λόγω ισοπαλίας.

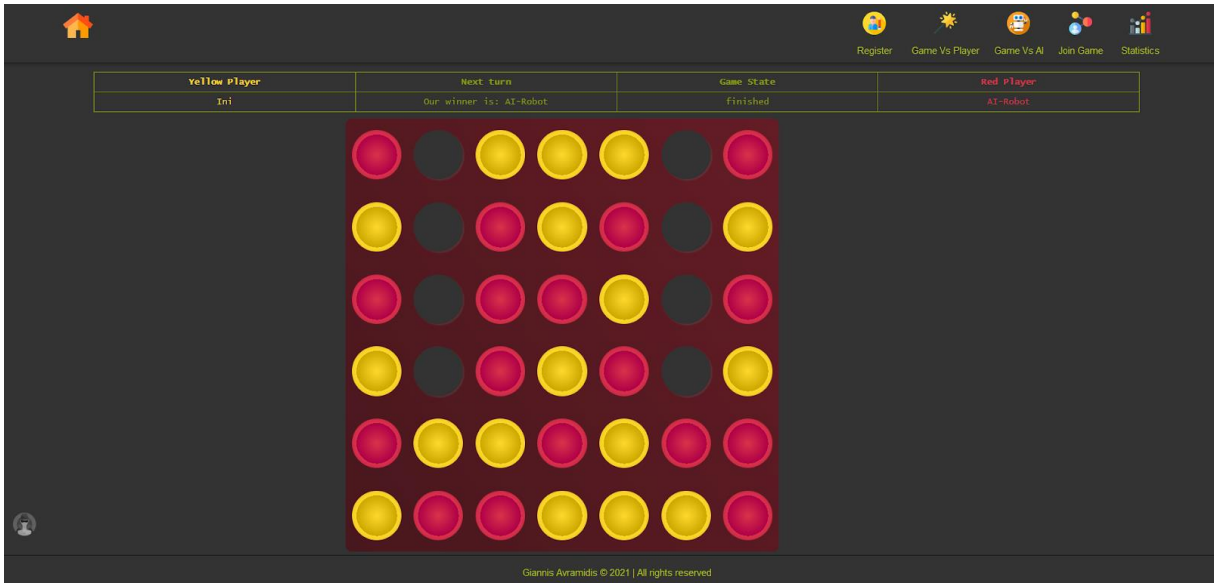


Σχήμα 15: Η κύρια σελίδα του παιχνιδιού (3)

Το παραπάνω στιγμιότυπα οθόνης είναι τραβηγμένο την στιγμή που έχει κερδίσει ο κίτρινος παίκτης το παιχνίδι. Βλέπουμε πάλι ότι έχει ανανεωθεί ο πίνακας στο πάνω μέρος της οθόνης. Παρατηρείται επίσης ότι το χρώμα του ταμπλό είναι αρχίζει να γίνεται κίτρινο (το χρώμα του παίκτη που κέρδισε). Το κίτρινο χρώμα αυξομειώνεται κάθε λίγα δευτερόλεπτα έχοντας διαγώνια κλίση.
Κάθε φορά που ένα παιχνίδι τελειώνει, γίνονται μια σειρά από ενέργειες σχετικές με τα στατιστικά των παικτών.

Αρχικά προστίθεται ένα παιχνίδι στα συνολικά παιχνίδια που έχει παίξει ο παίκτης και ανάλογα με την έκβαση του παιχνιδιού προστίθεται μια μονάδα είτε στον αριθμό των νικών είτε στον αριθμό των ισοπαλιών είτε στον αριθμό των ηττών του κάθε παίκτη. Τέλος, στην περίπτωση που το παιχνίδι έχει νικητή, προστίθεται 3 πόντοι στην συνολική βαθμολογία του νικητή και αφαιρούνται 3 πόντοι απο την συνολική βαθμολογία του ηττημένου.

4.9 Σελίδα παιχνιδιού εναντίον έξυπνου παίκτη



Σχήμα 16: Η σελίδα παιχνιδιού εναντίον του έξυπνου παίκτη

Η σελίδα του παιχνιδιού εναντίον του “έξυπνου” παίκτη είναι σχεδόν ίδια με την προηγούμενη σελίδα. Η κύριες διαφορές είναι ότι ο αντίπαλος παίκτης είναι συνεχώς ο AI-Robot και ότι οι κινήσεις του αντιπάλου γίνονται αυτόματα, με την βοήθεια του API που θα αναλυθεί στο επόμενο κεφάλαιο.

Τέλος μια ακόμα διαφορά έχει να κάνει με τις ενέργειες που γίνονται στο τέλος του παιχνιδιού. Σε περίπτωση νίκης αυξάνεται κατά μία μονάδα το πεδίο game_vs_ai του παίκτη.

4.10 Σελίδα εμφάνισης των error

Σχήμα 17: Η σελίδα εμφάνισης των error της εφαρμογής

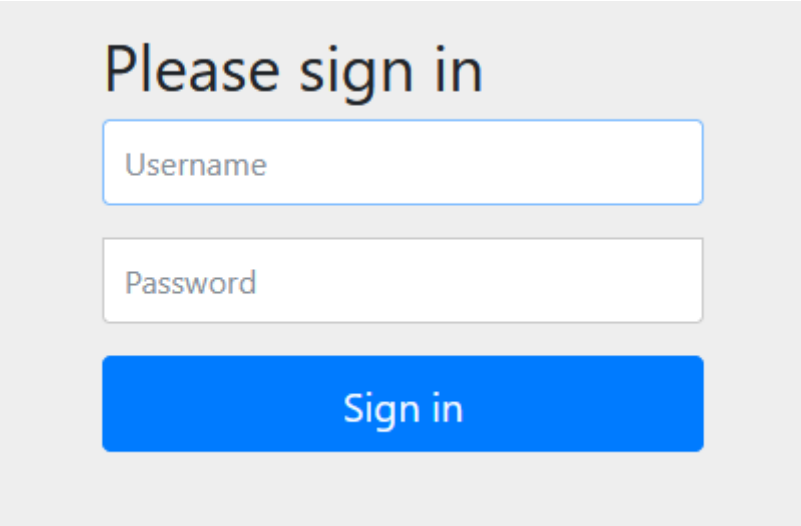
Όταν στην εφαρμογή προκύπτει κάποιο error, είτε από αυτά που έχουν προβλεφθεί προγραμματιστικά, είτε από αυτά που δεν μπορούν ή δεν έχουν προβλεφθεί, τότε εμφανίζεται μία σελίδα που έχει την αισθητική της παραπάνω εικόνας. Αναγράφει ως τίτλο το "Error page" συνοδευόμενο από το μήνυμα του error που έχει προκύψει ή από την έκφραση "unpredicted error" σε περίπτωση πού το error δεν έχει προβλεφθεί.

Στην πάνω δεξιά γωνία του μηνύματος υπάρχει ένα βελάκι το οποίο δείχνει προς τα αριστερά και όταν πατηθεί μεταφέρει το χρήστη στην προηγούμενη σελίδα για να ξανακάνει την ενέργεια που ήθελε παίρνοντας υπόψη το μήνυμα λάθους που έλαβε.

4.11 Σελίδα διαχείρισης της εφαρμογής

Η σελίδα διαχείρισης της εφαρμογής είναι ένα endpoint που παρέχεται από το Spring boot actuator και έχει συνδέσμους για όλα τα διαθέσιμα endpoint που παρέχονται. Έχουν αναλυθεί σε προηγούμενο κεφάλαιο οι δυνατότητες που προσφέρει. Περιληπτικά οι πιο βασικές είναι οι εξής:

- Παρακολούθηση της εφαρμογής μας
- Συλλογή μετρήσεων
- Παρακολούθηση της επισκεψιμότητας
- Παρακολούθηση της κατάστασης της βάσης δεδομένων.

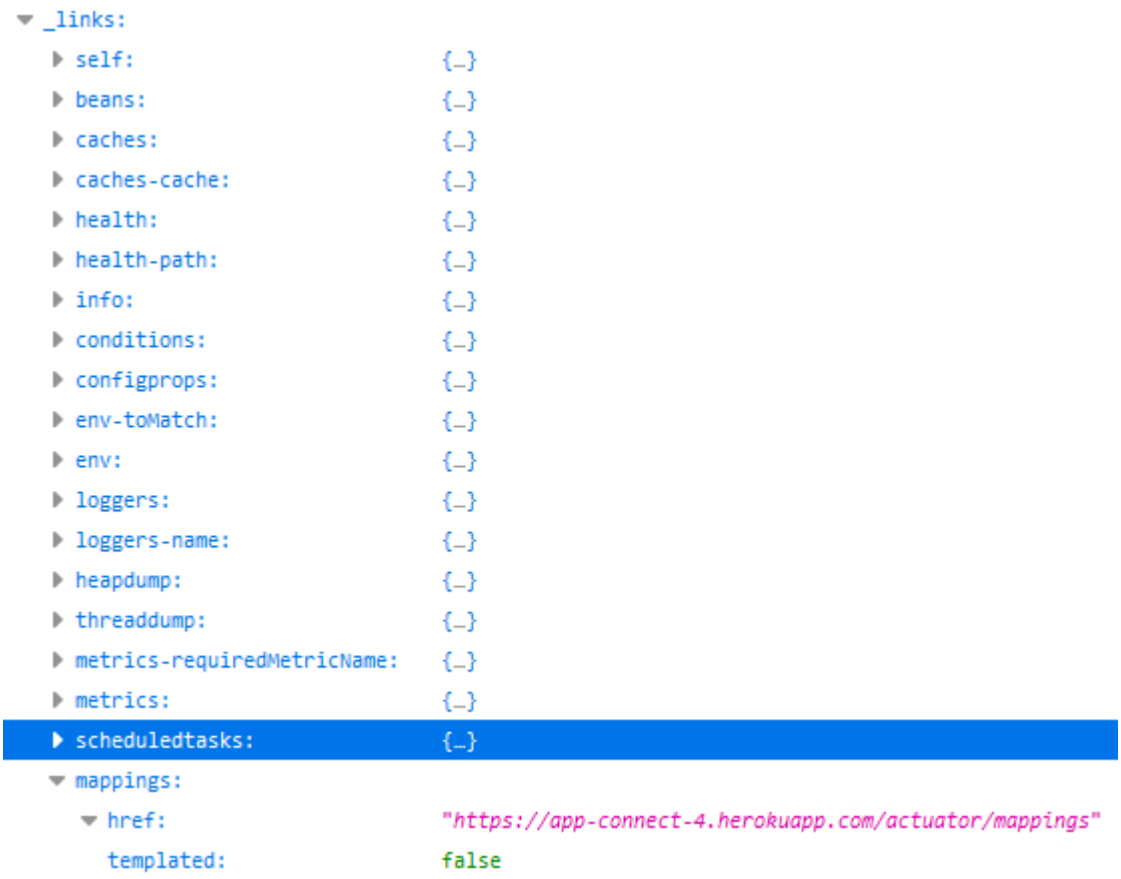


Σχήμα 18: Η φόρμα εισόδου για την σελίδα διαχείρισης της εφαρμογής

Αρκετά από τα endpoint του Spring Boot Actuator περιέχουν πληροφορίες οι οποίες είναι ευαίσθητες και θα πρέπει να είναι διαθέσιμα μόνο προς τον διαχειριστή της εφαρμογής. Για αυτόν τον λόγο, τα προαναφερόμενα endpoint είναι θωρακισμένα και για να γίνει χρήση αυτών θα πρέπει να γίνει επιτυχημένη εισαγωγή ονόματος χρήστη και κωδικού.

Το όνομα χρήστη και ο κωδικός είναι παραμετροποιημένα ώστε να ορίζονται εξωτερικά του προγράμματος. Συγκεκριμένα είναι 2 μεταβλητές περιβάλλοντος οι οποίες ορίζονται στον server που φιλοξενείται η εφαρμογή.

Έχοντας την δυνατότητα να αλλάζουμε συχνά κωδικό, χωρίς αυτό να απαιτεί re-deploy, καταφέρνουμε να κρατήσουμε την εφαρμογή πιο ασφαλή και πιο ευέλικτη.

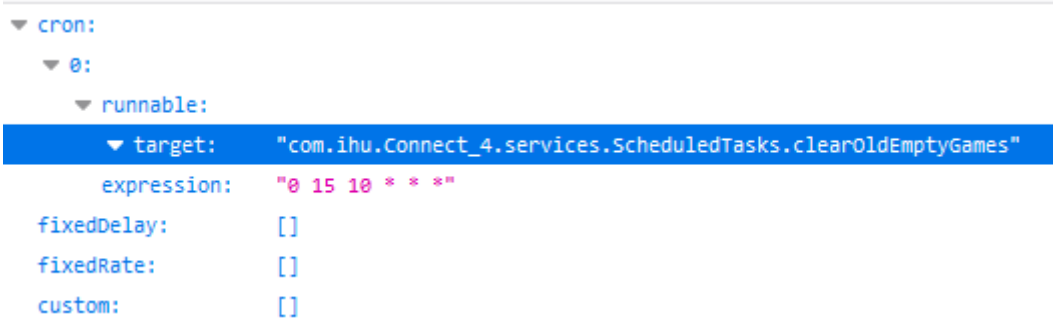


Σχήμα 19: Η σελίδα διαχείρισης της εφαρμογής

Το κύριο endpoint που παρέχεται από τον actuator βρίσκεται στο "/actuator". Τα endpoint ακολουθούν αρχιτεκτονική HATEOAS, δηλαδή παρέχουν links και πληροφορίες για τους διαφορετικούς "δρόμους" που μπορείς να ακολουθήσεις. Με αυτόν τον τρόπο διευκολύνεται πάρα πολύ η περιήγηση και η αλληλεπίδραση με ένα REST API.

Στο παραπάνω στιγμιότυπο οθόνης έχουν κρυφτεί τα links από τις διαθέσιμες επιλογές που έχουμε με σκοπό να χωρέσουν όλες στην εικόνα. Έχουμε αφήσει "ανοιχτή" την τελευταία διαθέσιμη ενέργεια με την ονομασία mappings για να γίνει κατανοητή η δομή που έχουν και οι υπόλοιπες.

4.12 Λοιπές λειτουργίες



Σχήμα 20: Η προγραμματισμένη διεργασία της εφαρμογής μέσα από την σελίδα διαχείρισης

Εκτός των βασικών λειτουργιών, υπάρχει και μια λειτουργία η οποία δεν παρέχεται στον χρήστη. Συγκεκριμένα είναι μια προγραμματισμένη διεργασία η οποία πραγματοποιείται μια φορά την ημέρα και είναι υπεύθυνη για την διαγραφή παλιών παιχνιδιών που δεν έχουν ολοκληρωθεί και κατ΄ επέκταση δεν επηρεάζουν ούτε τα στατιστικά των παικτών ούτε κάποια άλλη λειτουργία της εφαρμογής.

Ως παιχνίδια προς διαγραφή ορίζονται όσα έχουν δημιουργηθεί τουλάχιστον 40 λεπτά πριν και δεν έχουν ολοκληρωθεί ακόμα.

Κεφάλαιο 5ο: Παρουσίαση του API για τον «έξυπνο» παίκτη

5.1 Εισαγωγή

Στο παρόν κεφάλαιο θα περιγραφεί το API στο οποίο γίνονται οι αιτήσεις μέσω της εφαρμογής μου για να επιστρέφεται η καλύτερη δυνατή κίνηση με βάση τις ήδη καταχωρημένες κινήσεις των παικτών. Το API αναπτύχθηκε από τον Pascal Pons και ο τρόπος που το κατασκεύασε περιγράφεται στο blog του στο http://blog.gamesolver.org/ .

5.2 Η δομή του Connect 4 solver

Ακριβώς από κάτω παρατίθεται η περιγραφή του API όπως έχει γραφτεί από τον Pascal Prons (σε μετάφραση).
“Ορίζεται μια τιμή για οποιαδήποτε μη τελική θέση που αντικατοπτρίζει το αποτέλεσμα του παιχνιδιού για να παίξει ο παίκτης, δεδομένου ότι και οι δύο παίκτες παίζουν τέλεια και προσπαθούν να κερδίσουν το συντομότερο δυνατό ή να χάσουν όσο το δυνατόν αργότερα.
Μια θέση έχει:

- Θετική τιμή εάν ο τρέχων παίκτης μπορεί να κερδίσει. Συγκεκριμένα έχει την τιμή 1 αν μπορεί να κερδίσει με το τελευταίο του πούλι, την τιμή 2 αν μπορεί να κερδίσει με το προτελευταίο του πούλι και ούτω καθεξής.

- Μηδενική τιμή εάν το παιχνίδι θα λήξει ισοπαλία (δεδομένου ότι και οι 2 παίκτες παίζουν τις βέλτιστες κινήσεις).

- Αρνητική τιμή εάν ο τρέχων παίκτης χάνει ανεξαρτήτως της κίνησης που θα παίξει. Συγκεκριμένα έχει την τιμή -1 εάν ο αντίπαλός του κερδίζει με το τελευταίο του πούλι , -2 αν ο αντίπαλός του κερδίζει με το προτελευταίο του πούλι και ούτω καθεξής. “

5.3 Ο τρόπος χρήσης του API από την εφαρμογή μου και οι λειτουργίες που παρέχει

Επειδή οι περισσότερες δυνατότητες και πληροφορίες που προσφέρει το προαναφερόμενο API σχετίζονται με ένα παιχνίδι βέλτιστων κινήσεων, αποφάσισα να εξάγω και να χρησιμοποιήσω μόνο την πληροφορία της καλύτερης κίνησης.

Συγκεκριμένα αν παρατηρήσουμε τις πιθανές επιστρεφόμενες τιμές για τις θέσεις παιχνιδιού, μπορούμε να συμπεράνουμε ότι όσο μεγαλύτερη τιμή έχει μια θέση τόσο καλύτερη είναι στην δεδομένη χρονική στιγμή του παιχνιδιού (με εξαίρεση την τιμή 100 που δηλώνει ότι η συγκεκριμένη στήλη δεν είναι διαθέσιμη επειδή έχουν γεμίσει όλες οι θέσεις που έχει). Κατ’ επέκταση ελέγχοντας τις επιστρεφόμενες τιμές και βλέποντας σε ποια θέση βρίσκεται η μεγαλύτερη τιμή (εκτός του 100), ξέρουμε την καλύτερη κίνηση για τον παίκτη.

Οι λειτουργίες του κρυφού “cheat” κουμπιού και του παιχνιδιού εναντίον του “έξυπνου” παίκτη κάνουν χρήση αυτής της πληροφορίας. Συγκεκριμένα κάνουν αίτηση προς το API με τις κινήσεις που έχουν

παιχτεί στο παιχνίδι εκείνη την στιγμή, παίρνουν την επιστρεφόμενη πληροφορία εξάγουν την καλύτερη κίνηση και παίζουν την κίνηση αυτή στο ταμπλό, με την βοήθεια των ήδη υπαρχόντων μεθόδων και περνώντας τους ίδιους ακριβώς ελέγχους με μία απλή κίνηση.

5.4 Τεχνικές λεπτομέρειες για τις αιτήσεις προς το API

Για την υλοποίηση της αίτησης προς το API χρειάστηκε να κατασκευαστεί μια κλάση η οποία θα έχει πεδία που θα αντιστοιχούν στην αναμενόμενη απάντηση. Με αυτόν τον τρόπο είναι πιο εύκολο να αντιστοιχίζεται η απάντηση σε ένα Java αντικείμενο. Η αναμενόμενη απάντηση έχει την παρακάτω δομή:

```
{
 "type": "object",
 "properties": {
  "pos": {
   "type": "string"
  },
  "score": {
   "type": "array",
   "items": [
    {
     "type": "integer"
    },
    {
     "type": "integer"
    },
    {
     "type": "integer"
    },
    {
     "type": "integer"
    },
    {
     "type": "integer"
    },
    {
     "type": "integer"
    },
    {
     "type": "integer"
    },
    {
     "type": "integer"
    }
   ]
  }
```

```
  },
  "required": [
    "pos",
    "score"
  ]
}
```

Για αυτό κατασκευάστηκε ένα Data Transfer Object με την ονομασία CheatDTO το οποίο έχει δύο πεδία. Ένα String με την ονομασία pos και μία λίστα από integers με την ονομασία score.
Κάθε φορά που γίνεται μια αίτηση προς το API, η απάντηση, εφόσον ληφθεί χωρίς λάθη, μετατρέπεται σε αντικείμενο της κλάσης CheatDTO. Μετέπειτα μέσω κώδικα java πραγματοποιείται η διαδικασία που περιγράφηκε πιο πάνω:

- Αφαιρούνται όσες τιμές του πίνακα score έχουν τιμή μεγαλύτερη ίση του 100 (οι στήλες αυτές δεν είναι διαθέσιμες προς παίξιμο).

- Βρίσκουμε τον μεγαλύτερο αριθμό από τους υπόλοιπους.

- Επιστρέφουμε την θέση της τιμής αυτής στον πίνακα προσθέτοντας το 1 , γιατί οι διαθέσιμες στήλες του API είναι από 0 έως 6 ενώ στην συγκεκριμένη εφαρμογή είναι από 1 έως 7.

- Σε περίπτωση που υπάρξει κάποιο error κατά την διάρκεια της διαδικασίας αυτής, γίνεται throw και το αντίστοιχο exception.

Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις βελτίωσης

6.1 Συμπεράσματα

Στην παρούσα πτυχιακή εργασία, αναπτύχθηκε μια διαδικτυακή εφαρμογή με την οποία μπορεί ο χρήστης να παίξει το επιτραπέζιο σκορ 4, είτε εναντίον άλλων παικτών, είτε εναντίον ενός "έξυπνου" παίκτη. Για την υλοποίηση της χρησιμοποίησα πολλές ενδιαφέρουσες τεχνολογίες, κάποιες εκ των οποίων θεωρούνται de facto για την ανάπτυξη οποιασδήποτε διαδικτυακής εφαρμογής.

Η εξοικείωσή μου με αυτές τις τεχνολογίες και η βελτίωση μου στην συγγραφή κώδικα ήταν αναμφίβολα μεγάλο κέρδος για μένα.

Εκτός όμως των τεχνολογιών οι οποίες αλλάζουν και εξελίσσονται διαρκώς, θεωρώ σημαντικό κέρδος μου από αυτήν την πτυχιακή την τριβή μου με δομικά στοιχεία μιας web εφαρμογής όπως:

- Τη σχεδίαση του σχήματος μίας βάσης δεδομένων που θα καλύπτει όλες τις ανάγκες της εφαρμογής.
- Τη σύνδεση της με την εφαρμογή.
- Την επίλυση αλγοριθμικών προβλημάτων.
- Τη σωστή δόμηση ενός επεκτάσιμου και συντηρήσιμου προγράμματος ακολουθώντας ενδεδειγμένα σχεδιαστικά πρότυπα.
- Τη σχεδίαση και την υλοποίηση ενός REST API.
- Το documentation όλων των παραπάνω

6.2 Προτάσεις βελτίωσης

6.2.1 Εισαγωγή

Η επίλυση προβλημάτων και η προσθήκη νέων λειτουργιών είναι μια συνεχής διαδικασία στην ανάπτυξή εφαρμογών διότι μεγάλο μέρος αυτών εντοπίζονται ή και προκύπτουν μέσα από την χρήση τους. Έχοντας ως βάση την λογική αυτή ανέπτυξα την εφαρμογή με τρόπο που να διευκολύνει τον εντοπισμό λαθών, την επίλυσή τους και την προσθήκη νέων λειτουργιών.

Στα παρακάτω υποκεφάλαιο θα αναφερθούν τα βασικότερα προβλήματα προς επίλυση καθώς και κάποιες, κατά την γνώμη μου ενδιαφέρουσες ιδέες προς υλοποίηση.

6.2.2 Προτάσεις για επίλυση προβλημάτων

1. Στα παιχνίδια που παίζονται από πολλούς παίκτες ταυτόχρονα, ένα βασικό "πρόβλημα" είναι η άμεση ενημέρωση όλων των παικτών όταν συμβαίνει κάποια αλλαγή στην κατάσταση του παιχνιδιού. Ο τρόπος που το έχω προσεγγίσει στην εφαρμογή μου είναι ο ακόλουθος.
   Ο client κάνει περιοδικά αιτήσεις με την χρήση AJAX για να δει αν η κατάσταση του παιχνιδιού χρειάζεται ενημέρωση. Σε περίπτωση που χρειάζεται κάνει μια αίτηση για να λάβει τα δεδομένα

που χρειάζεται. Ο τρόπος αυτός δημιουργεί μικρές καθυστερήσεις στην εμφάνιση της κίνησης του αντιπάλου καθώς και πρόβλημα απόδοσης, συγκριτικά με κάποιους άλλους τρόπους υλοποίησης.
Για την επίλυση αυτού το προβλήματος καλό θα ήταν να γίνει χρήση τεχνολογιών που είτε επιτρέπουν την αμφίδρομη επικοινωνία μεταξύ client και server, είτε επιτρέπουν την ανεξάρτητη ειδοποίηση του client από τον server. Η πιο γνωστή τέτοια τεχνολογία είναι τα WebSocket, παρόλα αυτά καλό θα ήταν να γίνει μια έρευνα ανάμεσα και σε άλλες παρόμοιες τεχνολογίες για την επιλογή της καταλληλότερης.

2. Η ανάπτυξη της διεπαφής της εφαρμογής με την χρήση των Apache freemarker templates δημιούργησε κάποια προβλήματα σχετικά με την ανάπτυξη της διεπαφής σε σύνθετα ζητήματα και περιόρισε τις επιλογές που είχα.
   Έχοντας μια πιο ολοκληρωμένη εικόνα, θεωρώ ότι το front-end κομμάτι καλό θα ήταν να γίνει τελείως ανεξάρτητο και να υλοποιηθεί χωρίς την χρήση των template. Θα πρότεινα την χρήση ενός μοντέρνου και αξιόπιστου Javascript framework όπως είναι η Angular, η React ή το Vue.

3. Η Java σαν γλώσσα προγραμματισμού εκτός από τα πολλά θετικά για τα οποία είναι γνωστή, είναι γνωστή επίσης και για τον "περιττό" κώδικα που χρειάζεται να γραφτεί (boilerplate code).
   Η επικαιροποίηση της εφαρμογής κάνοντας χρήση της τελευταίας έκδοσης της Java (JDK 16) και των καινούργιων λειτουργιών που προσφέρει, καθώς και της βιβλιοθήκης lombok θα μπορούσε να μειώσει πολύ τον περιττό κωδικά και να διευκολύνει την περαιτέρω ανάπτυξη της εφαρμογής.

6.3 Προτάσεις για νέες λειτουργίες

1. Οι πληροφορίες που προσφέρει το εξωτερικό API που καλείται μέσω της εφαρμογής είναι πολλές. Θα μπορούσε με την χρήση αυτών να προστεθεί μια λειτουργία η οποία θα είχε εκπαιδευτικό χαρακτήρα και θα βοηθούσε στην προπόνηση με την βοήθεια του AI.

2. Η δυνατότητα επικοινωνίας των παικτών είτε μέσω chat στην κύρια σελίδα παιχνιδιού είτε μέσω της χρήσης emoticon θα βοηθούσε πολύ ώστε το παιχνίδι να γίνει πιο διαδραστικό και πιο ευχάριστο.

3. Από την αρχή δεν ήθελα η εγγραφή ενός παίκτη στην εφαρμογή να είναι μια χρονοβόρα διαδικασία. Πιστεύω ότι μια εφαρμογή για παιχνίδι είναι πιο ελκυστική όταν η δημιουργία και η έναρξη ενός παιχνιδιού είναι γρήγορη. Για την εγγραφή του παίκτη, στην εφαρμογή μου, απαιτείται μόνο ένα nickname. Παρόλα αυτά, χωρίς την χρήση κωδικού είναι λιγότερο αξιόπιστη η κατάταξη των παικτών.

   Θα πρότεινα μια εναλλακτική προσέγγιση που ο χρήστης θα έχει την δυνατότητα να εγγραφεί κανονικά (με χρήση κωδικού για να διασφαλίζεται ότι παίζει ο ίδιος παίκτης) και ταυτόχρονα θα υπάρχει η δυνατότητα κάποιος χρήστης να παίξει ως guest.

   Στην τελική κατάταξη των παικτών θα πρότεινα να λαμβάνονται υπόψη μόνο οι εγγεγραμμένοι παίκτες.

4. Η οριοθέτηση του χρόνου που έχει ο κάθε παίκτης για να παίξει θα βοηθούσε σημαντικά στην αποφυγή χαμένου χρόνου τόσο σε περιπτώσεις που κάποιος παίκτης αργεί όσο και σε περιπτώσεις που έχει αποσυνδεθεί οριστικά από το παιχνίδι. Καλό θα ήταν να ενημερωθεί ο πίνακας με τις πληροφορίες και να εμφανίζει και το χρόνο που υπολείπεται για τον παίκτη.

5. Μια ακόμη προσθήκη που θα είχε ενδιαφέρον, είναι η υλοποίηση και άλλων παιχνιδιών και η μετατροπή της εφαρμογής σε μια πλατφόρμα παιχνιδιών έτσι ώστε ο χρήστης να έχει περισσότερες δυνατότητες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

[1] Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra, *Head First Design Patterns*. O'Reilly Media, Inc., 2004.

[2] Joshua Bloch, *Effective Java*. Addison-Wesley Professional, 2017.

[3] Berners-Lee Tim, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. San Francisco: Harper, 2000.

[4] Kathy Sierra, Bert Bates, *Head First Java*. O'Reilly Media, Inc., 2005.

[5] Flanagan David, *JavaScript: The Definitive Guide*. O'Reilly Media Inc., 2006.

[6] Herman David, Effective JavaScript, Addison-Wesley Professional, 2013.

[7] Lethbridge C. Timothy, Laganiere Robert, *Μηχανική Αντικειμενοστραφούς Λογισμικού*. Εκδόσεις ΤΖΙΟΛΑΣ, 2016.

[8] Richardson Leonard, *RESTful Web APIs*. O'Reilly Media Inc., 2013.

[9] Randy Connolly, Ricardo Hoar, *Προγραμματισμός για το Web*. Εκδόσεις Μ. Γκιούρδας, 2015.

Internet Site

[10] w3scools.com, “CSS Tutorial”. [Online]. Available: https://www.w3schools.com/css/

[11] w3scools.com, “HTML Tutorial”. [Online]. Available: https://www.w3schools.com/html/

[12] Jet Brains, “IntelliJ IDEA”. [Online]. Available: https://www.jetbrains.com/idea/

[13] Maven, “Maven-Getting Started Guide”. [Online] Available: https://maven.apache.org/guides/getting-started/

[14] Spring, “Building REST services with Spring”. [Online] Available: https://spring.io/guides/tutorials/rest/

[15] Spring, “Securing a Web Application”. [Online] Available: https://spring.io/guides/gs/securing-web/