ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Visualize Zone: Εφαρμογή διαδικτύου για την οπτικοποίηση αλγορίθμων

Γεωργία Λουκία Β. Κοκκινού

Επιβλέποντες: **Δρ. Ιωάννης Χαμόδρακας,** Εργαστηριακό Διδακτικό Προσωπικό (ΕΔΙΠ)
Ιωάννης Εμίρης, Καθηγητής

ΑΘΗΝΑ

Μάρτιος 2020

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Visualize Zone: Εφαρμογή διαδικτύου για την οπτικοποίηση αλγορίθμων

Γεωργία Λουκία Β. Κοκκινού

Α.Μ.: 1115201400071

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Δρ. Ιωάννης Χαμόδρακας,** Εργαστηριακό Διδακτικό Προσωπικό (ΕΔΙΠ)
Ιωάννης Εμίρης, Καθηγητής

ΠΕΡΙΛΗΨΗ

Στα πλαίσια της παρούσας πτυχιακής εργασίας υλοποιήθηκε η εφαρμογή με όνομα Visualize Zone. Είχε ως στόχο την οπτικοποίηση αλγορίθμων της επιστήμης των υπολογιστών και της επιχειρησιακής έρευνας. Για το σκοπό αυτό χρησιμοποιήθηκε η βιβλιοθήκη Google OR Tools η οποία προσφέρει μία προγραμματιστική διεπαφή (API) για την επίλυση διαφόρων προβλημάτων. Η εφαρμογή αυτή προσφέρει στο χρήστη τη δυνατότητα ανεβάσματος αρχείων, οπτικοποίησης των αποτελεσμάτων και διαχείριση τους.

Κατά τη φάση της σχεδίασης υλοποιήθηκαν ορισμένα πρότυπα των σελίδων της εφαρμογής καθώς και μελετήθηκε ο σχεδιασμός με βάση το Material Design της Google. Επιλέχθηκαν τα κατάλληλα χρώματα για να επιτευχθεί η προσβασιμότητα της εφαρμογής καθώς και η σχεδίαση με βάση τις ανάγκες των χρηστών. Ακόμη δημιουργήθηκαν και ορισμένα διαγράμματα UML με βάση τα οποία γίνεται η μελέτη περιπτώσεων χρήσης που ήταν αρκετά πολύπλοκες όπως αυτή της πιστοποίησης του χρήστη για την είσοδο στην εφαρμογή ή της δημιουργίας ενός νέου πειράματος.

Στη φάση της υλοποίησης χρησιμοποιήθηκε η αρχιτεκτονική REST για την ανταλλαγή δεδομένων μεταξύ του νωτιαίου και του μετωπιαίου άκρου της εφαρμογής. Ο κώδικας στο νωτιαίο άκρο είναι γραμμένος με τη βοήθεια του Spring Boot Framework της JAVA ενώ στο μετωπιαίο άκρο ο κώδικας είναι σε Javascript ενώ έχει χρησιμοποιηθεί και το REACT js Framework για καλύτερη οργάνωση και επαναχρησιμοποίηση του κώδικα.

Το τελικό αποτέλεσμα αξιολογήθηκε με τη βοήθεια αρκετών αρχείων δεδομένων που εισήχθησαν στην εφαρμογή έτσι ώστε να γίνει η επεξεργασία τους και η οπτικοποίηση τους.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη Εφαρμογής Διαδικτύου

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μηχανική Λογισμικού,Οπτικοποίηση Δεδομένων, Επιχειρησιακή Έρευνα, React JS, Spring Boot

ABSTRACT

The purpose of this undergraduate thesis was the development of a web application named “Visualize Zone”, which executes and visualizes some well known algorithms in the field of Operational Research. For this purpose the Google OR Tools software library has been used to provide an API which solves some of that problems. The “Visualize Zone” application provides the capability of uploading files, storing information about various algorithms and visualizing the results.

During the design and gathering of functional requirements stages, some prototypes have been created using an online mockup tool. Additionally, the Material Design guidelines have been applied. The coloring of the application has been selected to promote accessibility for all users. Moreover, some UML diagrams have been made to analyze some complex use cases like the authorization and authentication process or the creation of a new experiment by the user.

In the development stage, a REST-based Architecture was selected for the communication and the exchange of data between the front end and the back end of the application. The back end API is implemented using the Spring Boot Framework whereas the front end is implemented in Javascript, using the React library in order to create optimized and reusable software code. The final result has been evaluated through the performance of tests with different datasets and algorithms.

SUBJECT AREA: Web Development

KEYWORDS: Software Engineering, Visualization, Operations Research, React JS, JAVA, Spring Boot

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα ήθελα να ευχαριστήσω τους επιβλέποντες μου Δρ. Ιωάννη Χαμόδρακα και Καθ. Ιωάννη Εμίρη για τη συνεργασία και την πολύτιμη συμβολή τους στην ολοκλήρωση της.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ 14

1. ΕΙΣΑΓΩΓΗ 14

1.1 Στόχος Εργασίας 14

1.2 Προεπισκόπηση κεφαλαίων 15

2. ΕΡΕΥΝΑ - ΒΙΒΛΙΟΘΗΚΕΣ ΛΟΓΙΣΜΙΚΟΥ 15

2.1 Η βιβλιοθήκη Google OR Tools 15

2.1.1 Linear Optimization 15

2.1.2 Routing 15

2.1.3 Packing 15

2.1.3.1 Knapsack 16

2.1.3.2 Multiple Knapsacks 16

3. ΣΧΕΔΙΑΣΜΟΣ 17

3.1 Χρήστες 17

3.2 Διαγράμματα UML 17

3.2.1 Διαγράμματα Περιπτώσεων Χρήσης 17

3.3 Material Design 19

3.3.1 Components του Material UI 20

3.3.2 Χρώματα και Material UI 20

3.4 Κανόνες Nielsen 21

3.5 Πρότυπα (Wireframes) 21

3.5.1 Σελίδες Εισόδου και Εγγραφής Χρήστη 21

3.5.2 Overview Εφαρμογής 22

3.5.3 Επιλογή Αλγορίθμου 23

3.5.4 Ανέβασμα Αρχείου 23

3.5.5 Γράφημα με Αρχικά Δεδομένα 24

3.5.6 Γράφημα για αποτελέσματα 25

3.5.7 Σελίδα ολοκλήρωσης πειράματος 25

3.5.8 Σελίδα ιστορικού πειραμάτων 26

3.5.9 Σελίδα Αποτελέσματος 26

3.5.10 Σελίδα Προφίλ - Αλλαγή Στοιχείων 27

3.5.11 Σελίδα Προφίλ - Αλλαγή Κωδικού 27

4. ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ 29

4.1 Νωτιαίο Άκρο Εφαρμογής 29

4.1.1 Spring Boot Framework 29

4.1.1.1 Πλεονεκτήματα 29

4.1.1.2 Μειονεκτήματα 29

4.1.1.3 Αρχείο application.properties 30

4.1.2 Maven 30

4.1.2.1 Βασικές αρχές του Maven 30

4.1.3 Πολυστρωματική Αρχιτεκτονική (Multi Layered Architecture) 32

4.1.4 MySQL 34

4.1.5 ORM και JPA 35

4.1.6 Spring Data, JPARepository και JPQL 37

4.1.7 REST και JSON 37

4.1.8 Authentication και Authorization με Spring Security 38

4.1.8.1 Annotations για Security 38
4.1.8.2 JWT 39
4.1.8.3 Φίλτρα σε HTTP Requests 40
4.1.8.4 hasAuthority και @PreAuthorize 40
4.2 Μετωπιαίο Άκρο 41
4.2.1 ReactJS 41
4.2.1.1 Χαρακτηριστικά του React 41
4.2.1.2 JSX 42
4.2.1.3 State και Props 43
4.2.1.4 React Components 43
4.2.2 Webpack 45
4.2.2.1 Πως λειτουργεί το webpack ? 46
4.2.2.2 ES6 και Transpiling 46
4.2.2.3 Παράδειγμα αρχείου webpack.config.js 46
4.2.3 React Vis Library 48
4.3 Δημοσίευση (Deployment) 50
4.3.1 Heroku 50
5. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ - ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ 52
5.1 Γενικά για τη δομή του κώδικα 52
5.2 Ανάλυση Αλγορίθμων 52
5.2.1 Αλγόριθμος Σακίδιου (Knapsack) 52
5.2.2 Αλγόριθμος Πολλαπλών Σακιδίων (Multiple Knapsacks) 52
5.2.3 Αλγόριθμος Περιπλανώμενου Πωλητή (Travelling Salesman) 52
5.2.4 Αλγόριθμος Γραμμικής Βελτιστοποίησης (Linear Optimization) 54
6. ΠΑΡΟΥΣΙΑΣΗ ΕΦΑΡΜΟΓΗΣ 55
6.1 Τελικό Αποτέλεσμα 55
6.2 Μελλοντικές Επεκτάσεις 55
6.3 Παρουσίαση Εφαρμογής 55
6.3.1 Αρχική Σελίδα 55
6.3.2 Σελίδα Εγγραφής 56
6.3.3 Σελίδα Εισόδου 56
6.3.4 Κεντρική Σελίδα (Overview) 57
6.3.5 Σελίδα Επιλογής Αλγορίθμου 57
6.3.6 Αλγόριθμος Σακιδίου (Knapsack) 58
6.3.6.1 Αρχείο για Αλγόριθμο Σακιδίου (Knapsack) 58
6.3.6.2 Ανέβασμα Αρχείου - Αρχικά Δεδομένα 59
6.3.6.3 Οπτικοποίηση Αρχικών δεδομένων (Knapsack) 59
6.3.6.4 Οπτικοποίηση Τελικού Αποτελέσματος (Knapsack) 60
6.3.7 Αλγόριθμος Πολλαπλών Σακιδίων (Multiple Knapsacks) 60
6.3.7.1 Μορφή αρχείου (Multiple Knapsacks) 60
6.3.7.2 Ανέβασμα Αρχείου - Αρχικά Δεδομένα 61
6.3.7.3 Οπτικοποίηση Αρχικών δεδομένων 62
6.3.7.4 Οπτικοποίηση Τελικού Αποτελέσματος 62
6.3.8 Αλγόριθμος Περιπλανώμενου Πωλητή (Travelling Salesman Problem) 63
6.3.8.1 Μορφή αρχείου (Travelling Salesman) 63
6.3.8.2 Οπτικοποίηση Τελικού Αποτελέσματος (Travelling Salesman) 63
6.3.9 Αλγόριθμος Γραμμικής Βελτιστοποίησης (Linear Optimization) 64
6.3.9.1 Μορφή αρχείου (Linear Optimization) 64

6.3.9.2 Οπτικοποίηση Τελικού Αποτελέσματος (Linear Optimization) 64
6.3.10 Αποθήκευση Αποτελέσματος 65
6.3.11 Αποθηκευμένα Δεδομένα 65
6.3.12 Εμφάνιση Πειραμάτων και κατέβασμα αρχικών και τελικών αρχείων 66
6.3.13 Σελίδα Προφίλ 66
ΣΥΜΠΕΡΑΣΜΑΤΑ 68
ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ 69
ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ 70
ΑΝΑΦΟΡΕΣ 71

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

Σχήμα 1: Use Case για σύνδεση χρήστη σελ.18

Σχήμα 2: Use Case για εγγραφή χρήστη σελ.18

Σχήμα 3: Use Case για δημιουργία πειράματος σελ.19

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1: Σήμα βιβλιοθήκης Google OR Tools σελ.15

Εικόνα 2: Σήμα Material Design σελ.19

Εικόνα 3: Επιλογή χρωμάτων σελ.21

Εικόνα 4: Πρότυπο - Εγγραφή Χρήστη σελ.22

Εικόνα 5: Πρότυπο - Σύνδεση Χρήστη σελ.22

Εικόνα 6: Πρότυπο - Αρχική Σελίδα σελ.23

Εικόνα 7: Πρότυπο - Επιλογή Αλγορίθμου σελ.23

Εικόνα 8: Πρότυπο - Ανέβασμα Αρχείου σελ.24

Εικόνα 9: Πρότυπο - Οπτικοποίηση Αρχικών Δεδομένων σελ.24

Εικόνα 10: Πρότυπο - Αποτελέσματα για TSP σελ.25

Εικόνα 11: Πρότυπο - Αποτελέσματα για Γραμμική Βελτιστοποίηση σελ.25

Εικόνα 12: Πρότυπο - Αποθήκευση Πειράματος σελ.26

Εικόνα 13: Πρότυπο - Αποθηκευμένα Πειράματα σελ.26

Εικόνα 14: Πρότυπο - Σελίδα Αποτελέσματος σελ.27

Εικόνα 15: Πρότυπο - Προφίλ Χρήστη σελ.27

Εικόνα 16: Πρότυπο - Αλλαγή Κωδικού Χρήστη σελ.28

Εικόνα 17: Spring Boot σελ.29

Εικόνα 18: Maven Lifecycle σελ.32

Εικόνα 19: Multi Layered Architecture σελ.33

Εικόνα 20: Δομή Κώδικα - Eclipse σελ.34

Εικόνα 21: Σχήμα Βάσης σελ.35

Εικόνα 22: JWT - Token σελ.39

Εικόνα 23: SPA vs Multi Page σελ.41

Εικόνα 24: React Lifecycle Methods σελ.45

Εικόνα 25: Webpack Transpiling σελ.46

Εικόνα 26: ES6 μετάφραση σε ES5 σελ.46

Εικόνα 27: React Vis - Scatter Plot σελ.49

Εικόνα 28: Heroku Logo σελ.50

Εικόνα 29: Δημοσίευση - Heroku σελ.51
Εικόνα 30: Αρχική Σελίδα σελ.56
Εικόνα 31: Εγγραφή σελ.56
Εικόνα 32: Σύνδεση Χρήστη σελ.57
Εικόνα 33: Κεντρική Σελίδα Εφαρμογής - Συνδεδεμένος Χρήστης σελ.57
Εικόνα 34: Επιλογή Αλγορίθμου σελ.58
Εικόνα 35: Ανέβασμα Αρχείου (Knapsack) σελ.59
Εικόνα 36: Οπτικοποίηση Αρχικών Δεδομένων - Knapsack σελ.60
Εικόνα 37: Οπτικοποίηση Αποτελεσμάτων - Knapsack σελ.60
Εικόνα 38: Ανέβασμα Αρχείου - Multiple Knapsacks σελ.62
Εικόνα 39: Οπτικοποίηση αρχικών δεδομένων - Multiple Knapsacks σελ.62
Εικόνα 40: Οπτικοποίηση αποτελεσμάτων - Multiple Knapsacks σελ.63
Εικόνα 41: Οπτικοποίηση Αποτελεσμάτων - Travelling Salesman Problem σελ.64
Εικόνα 42: Οπτικοποίηση Αποτελεσμάτων - Linear Optimization σελ.65
Εικόνα 43: Αποθήκευση Πειράματος σελ.65
Εικόνα 44: Διαχείριση Πειραμάτων σελ.66
Εικόνα 45: Εμφάνιση Αποτελέσματος - Multiple Knapsacks σελ.66
Εικόνα 46: Τροποποίηση Προσωπικών Δεδομένων Χρήστη σελ.67
Εικόνα 46: Αλλαγή Κωδικού σελ.67

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Πίνακας 1: Δεδομένα Εισόδου (Knapsack) σελ.58

Πίνακας 2: Δεδομένα Αρχείου - Multiple Knapsacks σελ.61

Πίνακας 3: Αρχείο Δεδομένων - Travelling Salesman σελ.63

Πίνακας 4: Αρχείο Δεδομένων - Linear Optimization σελ.64

ΠΡΟΛΟΓΟΣ

Η συγκεκριμένη εργασία διενεργήθηκε στο πλαίσιο ολοκλήρωσης προπτυχιακών σπουδών στο Τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών. Είχε ως στόχο την δημιουργία μιας εφαρμογής διαδικτύου με βάση την οποία οι χρήστες μπορούν να αποθηκεύουν και να οπτικοποιούν αποτελέσματα εκτέλεσης γνωστών αλγοριθμικών προβλημάτων τα οποία επιλύουν βιβλιοθήκες λογισμικού όπως η Google OR Tools.

Μερικές απο τις θεματικές περιοχές στις οποίες τα προβλήματα αυτά ανήκουν είναι η δρομολόγηση (Routing), η ομαδοποίηση δεδομένων σε κουτιά περιορισμένου μεγέθους (Packing) καθώς και η επίλυση εξισώσεων με γραμμικούς περιορισμούς (Linear Optimization).

Η εφαρμογή αυτή απευθύνεται κυρίως σε φοιτητές πληροφορικής, διδακτικό προσωπικό και σε όποιον θέλει να ασχοληθεί με πρακτικά προβλήματα που επιλύουν οι αλγόριθμοι που υλοποιεί η βιβλιοθήκη OR Tools.

Για την δημιουργία αυτής της εφαρμογής χρησιμοποιήθηκαν σύγχρονες τεχνολογίες διαδικτύου όπως είναι το Spring Boot Framework για την υλοποίηση του νωτιαίου άκρου της εφαρμογής (backend) και το Reactjs για το μετωπιαίο άκρο (front end). Στο πλαίσιο αυτού χρησιμοποιήθηκε σχεσιακή βάση δεδομένων MySQL καθώς και διάφορες ευκολίες που μας παρέχει το Spring Framework έτσι ώστε να διαχειριζόμαστε καλύτερα τη δομή της εφαρμογής και να απλοποιούμε την συντήρηση της. Ο κώδικας στο νωτιαίο άκρο ακολουθεί το σχεδιαστικό πρότυπο του τμήματος δεδομένων (Data Layer), τμήματος υπηρεσιών (Service Layer) και τμήματος παρουσίασης (Presentation Layer).

Το μετωπιαίο άκρο της εφαρμογής που υλοποιήθηκε με τη βοήθεια του React js αποτελείται από πολλά τμήματα (components) τα οποία συνθέτουν την τελική διεπαφή χρήστη (UI). Για την σύνδεση όλων αυτών τον τμημάτων και την διαχείριση των βιβλιοθηκών λογισμικού που χρησιμοποιήθηκαν γίνεται χρήση του εργαλείου webpack το οποίο είναι υπεύθυνο για την σύνθεση (transpiling) των αρχείων της εφαρμογής.

Κατά της υλοποίηση της αντιμετωπίστηκαν αρκετά προβλήματα που είχαν να κάνουν με τον μετασχηματισμό των δεδομένων τόσο από την πλευρά εισαγωγής από το χρήστη, όσο και από την πλευρά οπτικοποίησης τους από τη διεπαφή. Η επικοινωνία μεταξύ των δύο τμημάτων της εφαρμογής έγινε με τη χρήση του προτύπου REST καθώς και του JSON.

Το τελικό αποτέλεσμα της εφαρμογής είναι διαθέσιμο σε δημόσιο αποθετήριο του GitHub. Εκεί περιέχονται οδηγίες με τις εντολές για τον τρόπο μεταγλώττισης του προγράμματος.

Τέλος για την υλοποίηση της εφαρμογής ακολουθήθηκε και η τεχνική της συνεχούς δημοσίευσης της εφαρμογής (Continuous Deployment) στο Heroku που αποτελεί μία πλατφόρμα υπολογιστικού νέφους (Cloud).

1. ΕΙΣΑΓΩΓΗ

1.1 Στόχος Εργασίας

Στόχος της συγκεκριμένης εργασίας είναι η δημιουργία μιας εφαρμογής διαδικτύου η οποία θα προσφέρει στους χρήστες τη δυνατότητα να δοκιμάσουν και να οπτικοποιήσουν τα αποτελέσματα ορισμένων αλγοριθμικών προβλημάτων.
Για να επιτευχθεί το συγκεκριμένο αποτέλεσμα έγινε χρήση του Spring Boot για το backend της εφαρμογής καθώς και του React για το μετωπιαίο άκρο (frontend). Επιπλέον η εφαρμογή είναι ανεβασμένη στο PaaS (Platform as a Service) Heroku. Η συγκεκριμένη εργασία συνδυάζει δύο πεδία της σύγχρονης πληροφορικής την ανάπτυξη και σχεδίαση λογισμικού καθώς και τη διαχείριση δεδομένων και γνώσης.

1.2 Προεπισκόπηση κεφαλαίων

Η εργασία είναι χωρισμένη σε έξι μέρη. Το πρώτο μέρος μελετά τη χρήση της βιβλιοθήκης Google OR Tools καθώς και την έρευνα γύρω από τις υπόλοιπες βιβλιοθήκες και τα εργαλεία λογισμικού που χρησιμοποιήθηκαν για την υλοποίηση της συγκεκριμένης εργασίας. Στο δεύτερο μέρος της γίνεται ανάλυση του σχεδιασμού με διαγράμματα UML ορισμένων περιπτώσεων χρήσης καθώς και αναλύεται η επιλογή του Material Design ως σχεδιαστικού προτύπου. Στο τρίτο μέρος αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν, οι τεχνικές και οι βιβλιοθήκες λογισμικού που βοήθησαν στη δημιουργία του τελικού αποτελέσματος. Αναλύεται η δομή τόσο του νωτιαίου άκρου της εφαρμογής καθώς και του μετωπιαίου, το σχήμα της βάσης, ο τρόπος δημοσίευσης της εφαρμογής (deployment). Επιπλέον στο πέμπτο μέρος του παρόντος εγγράφου περιγράφεται το API που δημιουργήθηκε από την πλευρά του backend για την επίλυση των αλγοριθμικών προβλημάτων, η χρήση της βιβλιοθήκης Google OR Tools στην πράξη καθώς και οι συναρτήσεις και οι κλάσεις που το συνθέτουν. Τέλος, στο έκτο μέρος γίνεται επίδειξη του τελικού αποτελέσματος και παρουσιάζονται και ορισμένες προτάσεις για βελτίωση και επέκταση του. Επιπλέον υπάρχει ευρετήριο των εικόνων και των σχημάτων που χρησιμοποιήθηκαν καθώς και πίνακας με την ορολογία και τα ακρωνύμια.

2. ΕΡΕΥΝΑ - ΒΙΒΛΙΟΘΗΚΕΣ ΛΟΓΙΣΜΙΚΟΥ

2.1 Η βιβλιοθήκη Google OR Tools

Η βιβλιοθήκη Google OR Tools είναι μία βιβλιοθήκη ανοιχτού κώδικα που έχει ως στόχο την επίλυση προβλημάτων βελτιστοποίησης. Είναι σχεδιασμένη ώστε να προσφέρει μία διεπαφή για τον προγραμματιστή (API) το οποίο είναι γραμμένο σε γνωστές γλώσσες προγραμματισμού όπως είναι η C++, Python, C#, Java. Προσφέρει έτοιμες συναρτήσεις για την επίλυση προβλημάτων γραμμικού προγραμματισμού (linear programming), προγραμματισμού ικανοποίησης περιορισμών (CSP), δρομολόγησης (routing) όπως και δύσκολα επιλύσιμα προβλήματα (NP - Hard) όπως το πρόβλημα του σακιδίου (Knapsack). Ο προγραμματιστής το μονο που έχει να κάνει είναι να επιλέξει κάποιους επιλυτές αυτών τον προβλημάτων καθώς και να το μοντελοποιήσει δίνοντας τους περιορισμούς και το πεδίο ορισμού.
Για το σκοπό αυτό δίνονται αρκετά παραδείγματα κώδικα για την μοντελοποίηση των προβλημάτων στην ιστοσελίδα της βιβλιοθήκης.

Εικόνα 1: Σήμα βιβλιοθήκης Google OR Tools

2.1.1 Linear Optimization

Ένα από τα προβλήματα τα οποία μοντελοποιούνται και οπτικοποιούνται στην παρούσα εργασία είναι αυτό της γραμμικής βελτιστοποίησης. Ανήκει στην κατηγορία προβλημάτων που μελετά ο γραμμικός προγραμματισμός και είναι προβλήματα των οποίων οι παράμετροι (variables) και οι περιορισμοί (constraints) εκφράζονται με γραμμικές σχέσεις. Η βιβλιοθήκη OR Tools χρησιμοποιεί τον λύτη (solver) Glop για την επίλυση τέτοιων προβλημάτων. Δίνονται παραδείγματα απλής επίλυσης ορισμένων προβλημάτων για παράδειγμα με γραμμικές εξισώσεις και ανισότητες στην ιστοσελίδα της βιβλιοθήκης.

2.1.2 Routing

Ίσως ένα από τα πιο ενδιαφέροντα προβλήματα που μελετά η επιχειρησιακή έρευνα είναι η δρομολόγηση (Routing). Ένα υποπρόβλημα αυτής της κατηγορίας είναι το πρόβλημα του περιπλανώμενου πωλητή (Travelling Salesman Problem). Σύμφωνα με την περιγραφή του προβλήματος ένας πωλητής προσπαθεί να διανύσει την μικρότερη απόσταση έτσι ώστε να διανέμει τα προϊόντα του σε όλες τις πόλεις. Ο υπολογισμός όλων των πιθανών μονοπατιών για μικρά σύνολα πόλεων είναι σχετικά γρήγορος όμως καθώς αυξάνεται το σύνολο, αυξάνεται και η χρονική πολυπλοκότητα εκθετικά. Γι’αυτό το λόγο χρειάζεται μοντελοποίηση του συνόλου των πόλεων σε έναν γράφο έτσι ώστε να εφαρμοστούν διάφορες μέθοδοι βελτιστοποίησης για να βρεθεί η καλύτερη δυνατή λύση. Ο συγκεκριμένος γράφος έχει ως κόμβους τις πόλεις ενώ οι ακμές του είναι η απόσταση μεταξύ των πόλεων. Τέλος για την επίλυση του χρησιμοποιείται μία δομή που λέγεται πίνακας αποστάσεων (distance matrix).

2.1.3 Packing

Στόχος των προβλημάτων πακεταρίσματος (packing problems) είναι να βρεθεί ο καλύτερος τρόπος με το οποίο μπορούν να ομαδοποιηθούν αντικείμενα σε διαφορετικούς

κάδους με συγκεκριμένη χωρητικότητα έτσι ώστε να μένουν τα λιγότερα δυνατά έξω από τους κάδους.

2.1.3.1 Knapsack

Στο κλασικό πρόβλημα του σακιδίου υπάρχει μόνο ένα δοχείο με προκαθορισμένο μέγεθος στο οποίο συνήθως χωράει ένα υποσύνολο των αντικειμένων ανάλογα με τα βάρη τους. Γενικότερα στο συγκεκριμένο πρόβλημα θέλουμε να βελτιστοποιήσουμε το άθροισμα των βαρών των αντικειμένων ώστε να χωράει στο σακίδιο μας.

2.1.3.2 Multiple Knapsacks

Μία παραλλαγή του προβλήματος του σακιδίου είναι το πρόβλημα των πολλαπλών σακιδίων. Σε αυτή την περίπτωση θέλουμε να βελτιστοποιήσουμε τα αντικείμενα που χωρούν σε περισσότερους από έναν σάκους με διαφορετική χωρητικότητα.

3. ΣΧΕΔΙΑΣΜΟΣ

3.1 Χρήστες

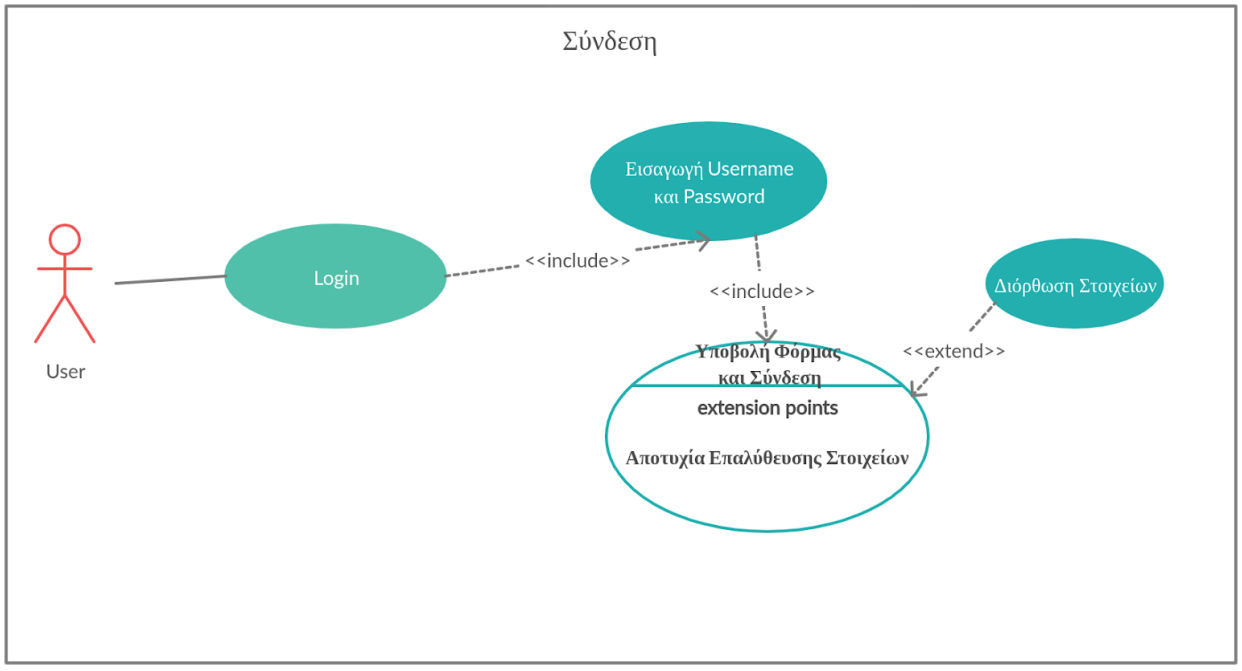
Το προφίλ των χρηστών της εφαρμογής σύμφωνα με την έρευνα άλλων παρόμοιων εφαρμογών που οπτικοποιούν γνωστούς αλγορίθμους αποτελείται από φοιτητές, διδακτικό προσωπικό και όποιον θέλει να έρθει σε επαφή με την επιστήμη των υπολογιστών.

3.2 Διαγράμματα UML

Ένας τρόπος με τον οποίο μπορεί να επιτευχθεί ο καλύτερος σχεδιασμός του συστήματος είναι η χρήση της ενιαίας γλώσσας μοντελοποίησης (UML) για τη μοντελοποίηση των λειτουργικών απαιτήσεων μέσω διαγραμμάτων. Η UML χρησιμοποιείται για την τεκμηρίωση και την περιγραφή ενός συστήματος από τους προγραμματιστές, τους διαχειριστές ενός έργου λογισμικού, τους ανθρώπους που τεστάρουν την ποιότητα του προϊόντος και κάθε εμπλεκόμενου προσώπου. Υπάρχουν πολλά είδη διαγραμμάτων που χρησιμοποιούνται για την τεκμηρίωση του σχεδιασμού όπως είναι τα διαγράμματα κλάσεων, τα διαγράμματα περιπτώσεων χρήσης, τα διαγράμματα καταστάσεων, ακολουθίας και πακέτων. Παρακάτω θα δούμε μερικά διαγράμματα περιπτώσεων χρήσης που χρησιμοποιήθηκαν για την μοντελοποίηση ορισμένων πολύπλοκων περιπτώσεων χρήσης.

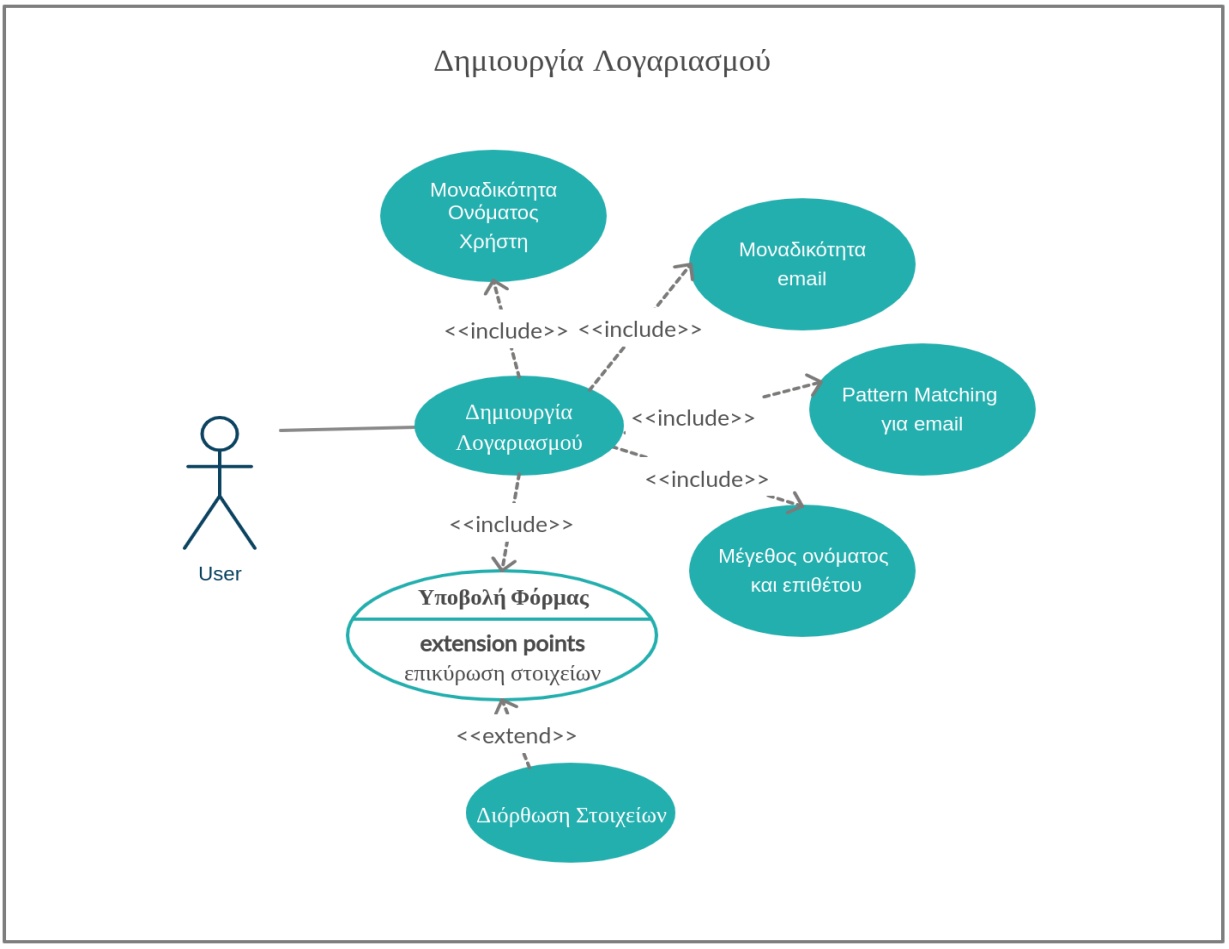
3.2.1 Διαγράμματα Περιπτώσεων Χρήσης

Τα διαγράμματα περιπτώσεων χρήσης (Use Case Diagrams) αποτελούν έναν τρόπο απεικόνισης της αλληλεπίδρασης των χρηστών της εφαρμογής με το σύστημα. Τα σχήματα που συναντάμε σε ένα διάγραμμα περιπτώσεων χρήσης είναι η **έλλειψη** η οποία περιέχει την ενέργεια που θέλουμε να επιτύχουμε και τα **βέλη** που χρησιμοποιούνται για να δηλώσουν τη σχέση μεταξύ δύο ενεργειών. Υπάρχουν δύο σχέσης μεταξύ ενεργειών που μπορούν να οπτικοποιηθούν με τη βοήθεια του βέλους 1) η σχέση συμπερίληψης (include) η οποία δηλώνει ότι η μία ενέργεια προϋποθέτει την εκτέλεση της άλλης πρώτα και 2) η σχέση επέκτασης (extends) που επεκτείνει την ενέργεια π.χ. στην περίπτωση λάθους.
Παρακάτω βλέπουμε ένα διάγραμμα το οποίο επεξηγεί την διαδικασία εισόδου ενός εγγεγραμμένου χρήστη:



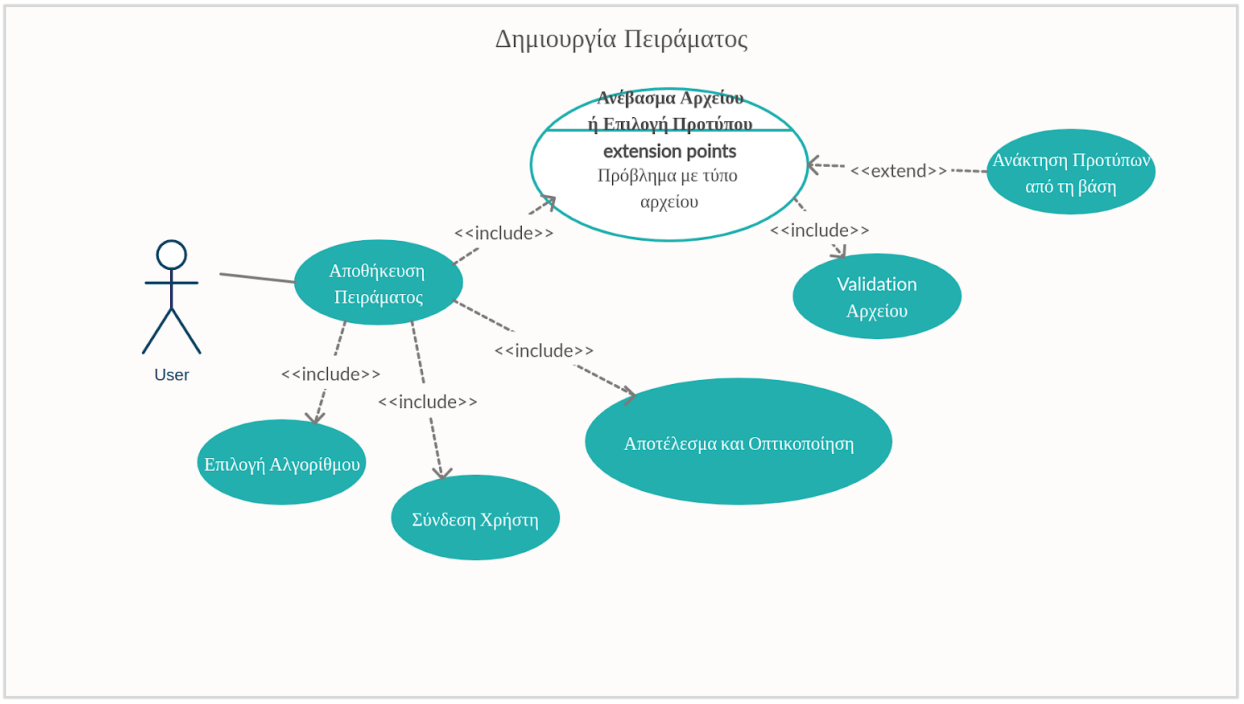
Σχήμα 1: Use Case για σύνδεση χρήστη

Το επόμενο διάγραμμα περιγράφει τη διαδικασία εγγραφής ενός νέου χρήστη στο σύστημα:



Σχήμα 2: Use Case για εγγραφή χρήστη

Τέλος έχουμε ακόμη ένα διάγραμμα που περιγράφει ολόκληρη την διαδικασία δημιουργίας ενός νέου πειράματος με τον αλγόριθμο που έχει επιλέξει ο χρήστης:



Σχήμα 3: Use Case για δημιουργία πειράματος

3.3 Material Design

Για το σχεδιασμό της εφαρμογής χρησιμοποιήθηκε η σχεδιαστική γλώσσα της google που ονομάζεται Material Design. Οι περισσότερες δημοφιλείς εφαρμογές της google χρησιμοποιούν την συγκεκριμένη σχεδίαση όπως είναι το Google Maps, το Google Drive ή το Gmail.

Εικόνα 2: Σήμα Material Design

Το Material Design όπως και κάθε σχεδιαστικό ρεύμα για την υλοποίηση εφαρμογών έχει ορισμένες αρχές. Κάποιες από αυτές είναι:

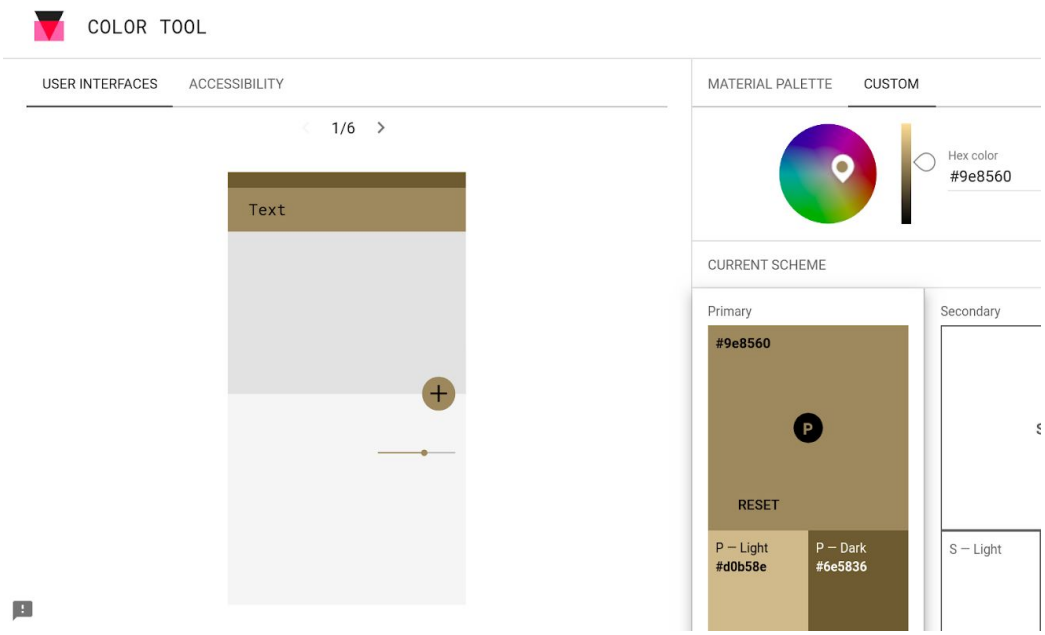
- **Το υλικό είναι η μεταφορά (Material is the metaphor):** Η ανάπτυξη του συγκεκριμένου σχεδιαστικού προτύπου είναι εμπνευσμένη από τα χειροπιαστά αντικείμενα τα οποία χρησιμοποιούμε καθημερινά όπως είναι το χαρτί και το μελάνι. Αυτό κάνει τα αντικείμενα της διεπαφής χρήστη να ξεχωρίζουν καλύτερα το ένα από το άλλο καθώς χρησιμοποιούν συμβολισμούς που συναντάμε στα κλασικά αντικείμενα της τυπογραφίας.
- **Η κίνηση προσφέρει νόημα (Motion provides meaning):** Στο material design πρέπει να έχουμε κίνηση και μετασχηματισμούς τον αντικειμένων της διεπαφής που προσφέρουν κάποιο νόημα στην εφαρμογή μας καθώς και ειδοποιούν τον χρήστη για κάποια αλλαγή.

3.3.1 Components του Material UI

Το Material Design έχει ως στόχο να βοηθήσει τον προγραμματιστή να σχεδιάσει εφαρμογές που να ανταποκρίνονται σε διαφορετικά μεγέθη οθονών (responsive design). Ορισμένα από αυτά είναι τα εξής: μπάρες για την πλοήγηση (App bars), πίνακες δεδομένων (data tables), καρτέλες (tabs), κάρτες (cards), μπάρα πλοήγησης (navigation drawer) όπως και γραμμικοί δείκτες προόδου (linear progress bars).
Υπάρχουν διάφορες υλοποιήσεις αυτών των αντικειμένων όπως είναι το Polymer καθώς και το Material-UI για το React Framework που χρησιμοποιήθηκε για την παρούσα υλοποίηση.

3.3.2 Χρώματα και Material UI

Ακόμη ένα χαρακτηριστικό του σχεδιασμού με βάση το πρότυπο Material Design είναι και η επιλογή των χρωμάτων των αντικειμένων της διεπαφής χρήστη. Συνήθως χρησιμοποιείται μία μονοχρωματική παλέτα η οποία περιλαμβάνει ένα κυρίως (primary) χρώμα, ένα δευτερεύον (secondary) χρώμα καθώς και διάφορες σκουρότερες ή πιο ανοιχτές αποχρώσεις του ίδιου χρώματος. Επιπλέον είναι σημαντικό το ότι προτείνονται τα χρώμα των γραμμάτων που πρέπει να χρησιμοποιηθούν ανάλογα με το χρώμα του αντικειμένου ή το φόντο. Αυτό γίνεται για λόγους προσβασιμότητας έτσι ώστε ο σχεδιαστής να επιλέξει μεταξύ του λευκού ή του μαύρου. Για να επιλεγούν τα χρώματα της παλέτας για την εφαρμογή χρησιμοποιήθηκε ένα εργαλείο σχεδιασμού που προσφέρεται δωρεάν από το Material Design. Τα αποτελέσματα αυτού ήταν τα εξής:



Εικόνα 3: Επιλογή χρωμάτων

Αρχικά επιλέγουμε ένα από τα χρώματα που θέλουμε να χρησιμοποιήσουμε και έπειτα το εργαλείο μας δίνει και άλλα της ίδιας μονοχρωματικής παλέτας.

3.4 Κανόνες Nielsen

Κατά την δημιουργία των προτύπων (wireframes) της εφαρμογής αυτής χρησιμοποιήθηκαν και οι κανόνες τους Nielsen που βοηθούν στην ευρετική αξιολόγηση ενός σχεδιασμού. Οι συγκεκριμένοι κανόνες είναι οι παρακάτω:

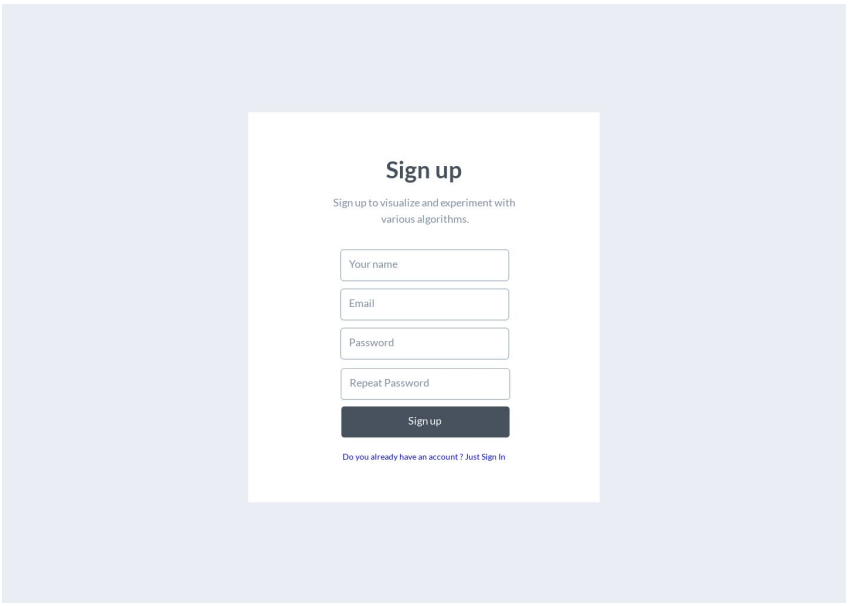
1. Ορατότητα της κατάστασης του συστήματος (Visibility)
2. Ομοιότητα/αναλογία με πραγματικό κόσμο (Match)
3. Δυνατότητα ελέγχου (Control)
4. Συνέπεια (Consistency)
5. Πρόληψη Λαθών (Prevention)
6. Αναγνώριση έναντι επανάκτησης (Recognition)
7. Ευελιξία (Flexibility)
8. Μινιμαλιστική Σχεδίαση (Minimalism)
9. Δυνατότητα Ανάκαμψης (Recovery)
10. Βοήθεια (Help)

3.5 Πρότυπα (Wireframes)

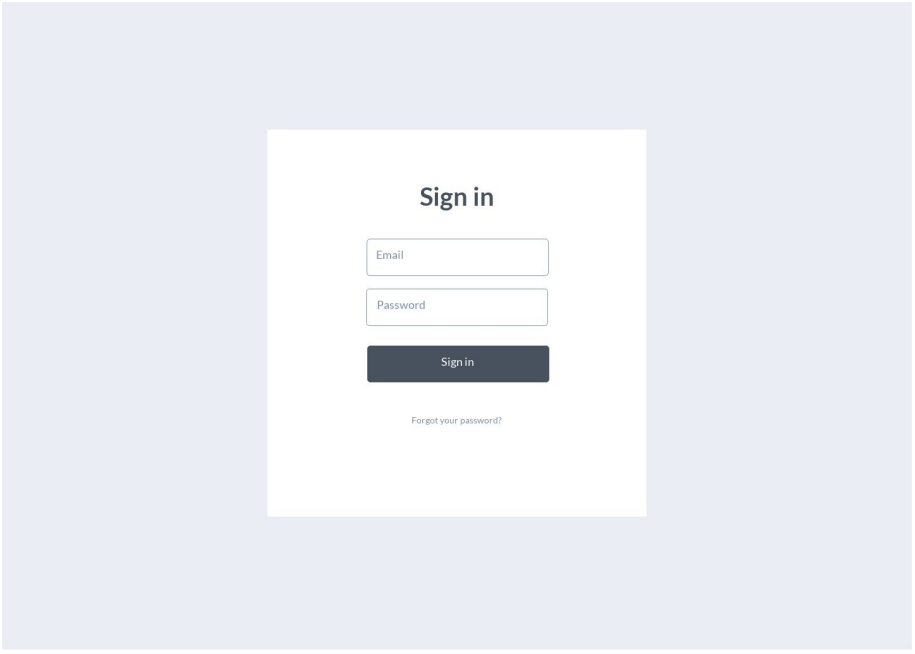
Τα πρότυπα της εφαρμογής σε μεσαία πιστότητα δημιουργήθηκαν με τη βοήθεια της εφαρμογής MarvelApp η οποία προσφέρει αρκετά εργαλεία για το σχεδιασμό.

3.5.1 Σελίδες Εισόδου και Εγγραφής Χρήστη

Στα παρακάτω wireframes παρουσιάζεται η σελίδα εγγραφής και εισόδου του χρήστη στην εφαρμογή.



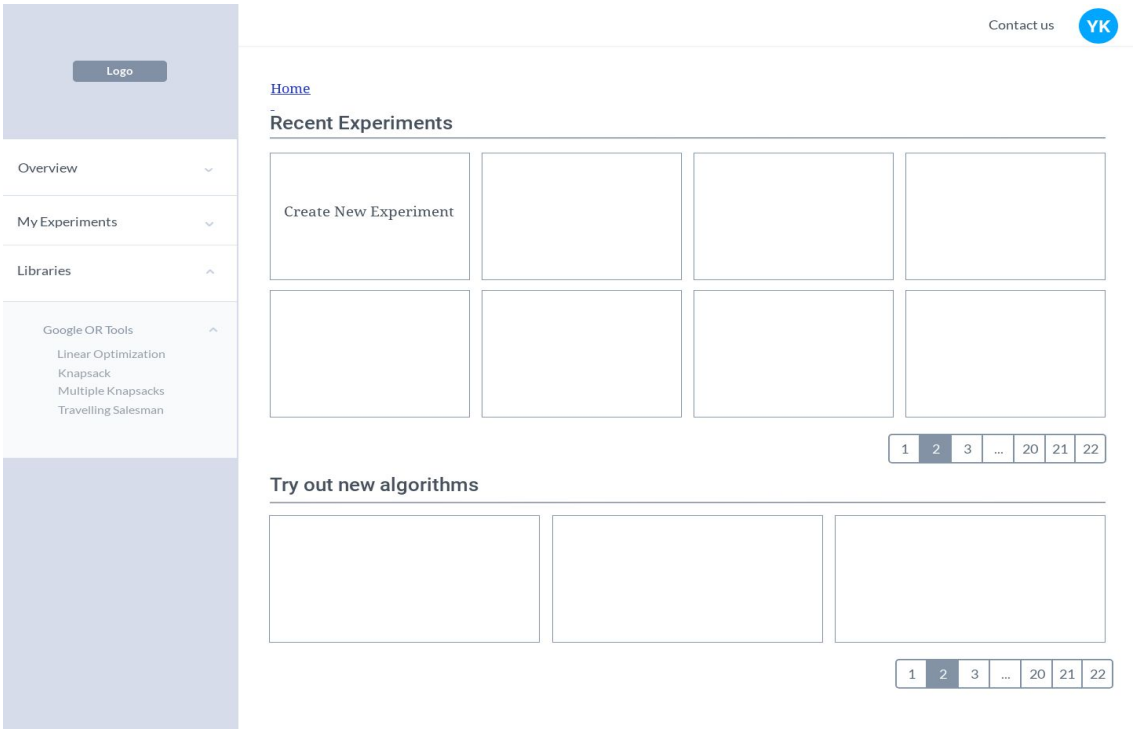
Εικόνα 4: Πρότυπο - Εγγραφή Χρήστη



Εικόνα 5: Πρότυπο - Σύνδεση Χρήστη

3.5.2 Overview Εφαρμογής

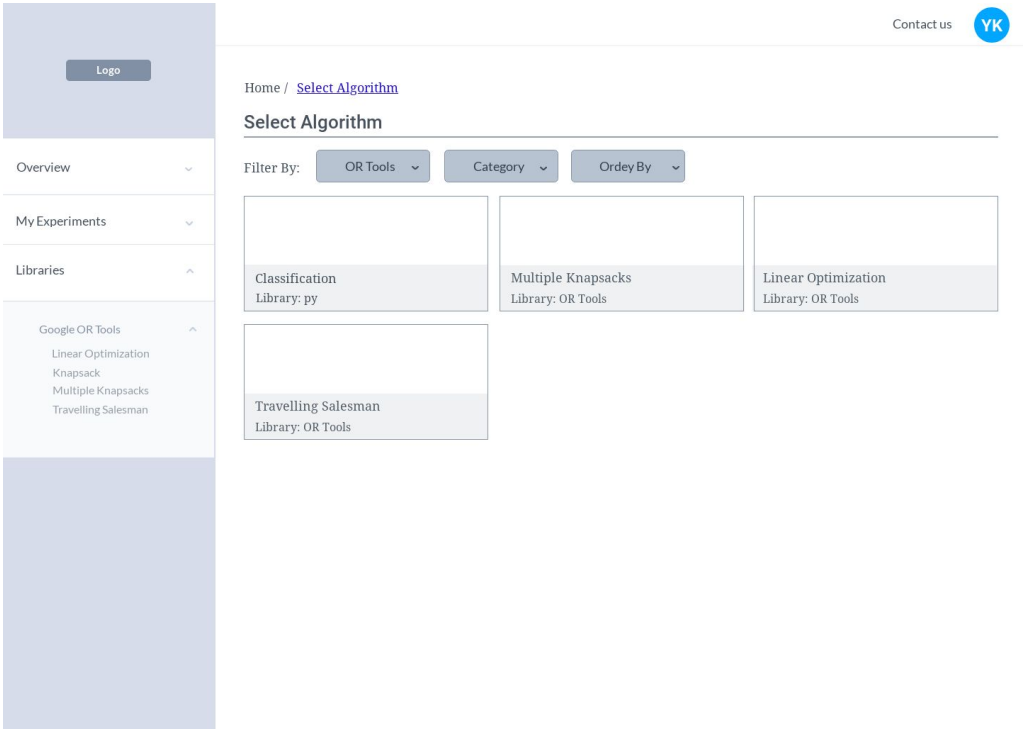
Σε αυτό το πρότυπο παρουσιάζεται η κεντρική σελίδα της εφαρμογής. Έχουν χρησιμοποιηθεί διάφορα αντικείμενα του material design όπως είναι οι κάρτες (cards) για την παρουσίαση των πρόσφατων πειραμάτων που έχουν γίνει από το χρήστη καθώς και η μπάρα πλοήγησης (drawer).



Εικόνα 6: Πρότυπο - Αρχική Σελίδα

3.5.3 Επιλογή Αλγορίθμου

Στο παρακάτω wireframe παρουσιάζεται το μενού επιλογής αλγορίθμου. Ο χρήστης μπορεί να φιλτράρει με βάσει την κατηγορία του αλγορίθμου (π.χ. packing) ή τη βιβλιοθήκη (π.χ. OR Tools).

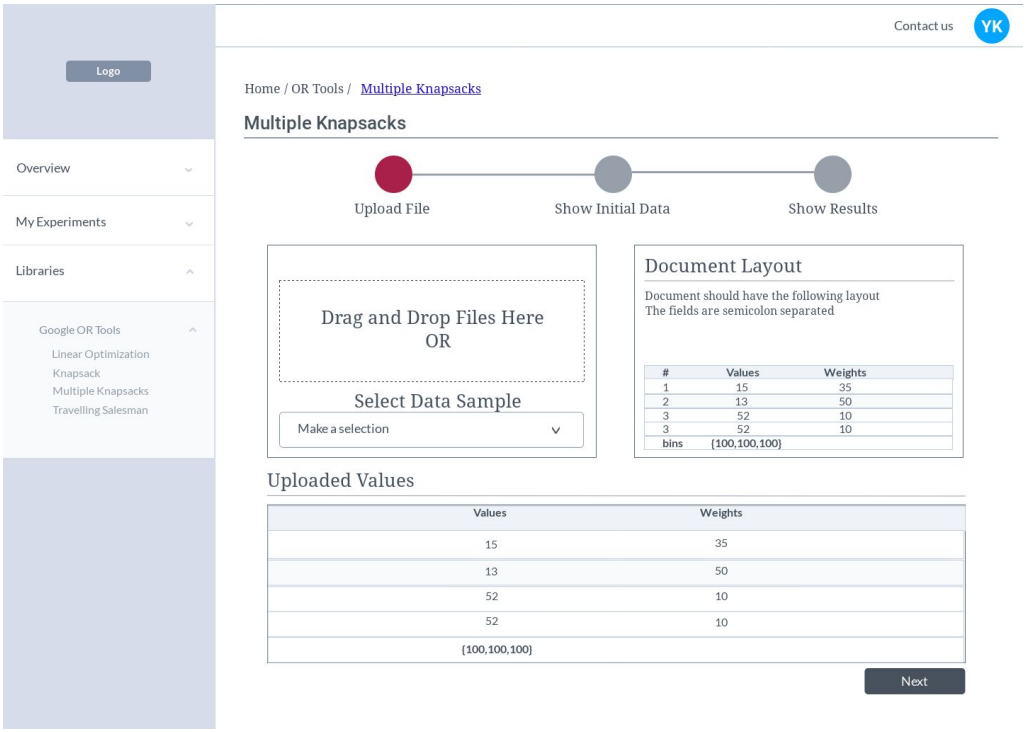


Εικόνα 7: Πρότυπο - Επιλογή Αλγορίθμου

3.5.4 Ανέβασμα Αρχείου

Σε αυτό το wireframe βλέπουμε τη σελίδα που χρησιμοποιείται για το ανέβασμα του αρχείου. Επιπλέον δίνεται ένας πίνακας με οδηγίες για την μορφή που πρέπει να έχει το

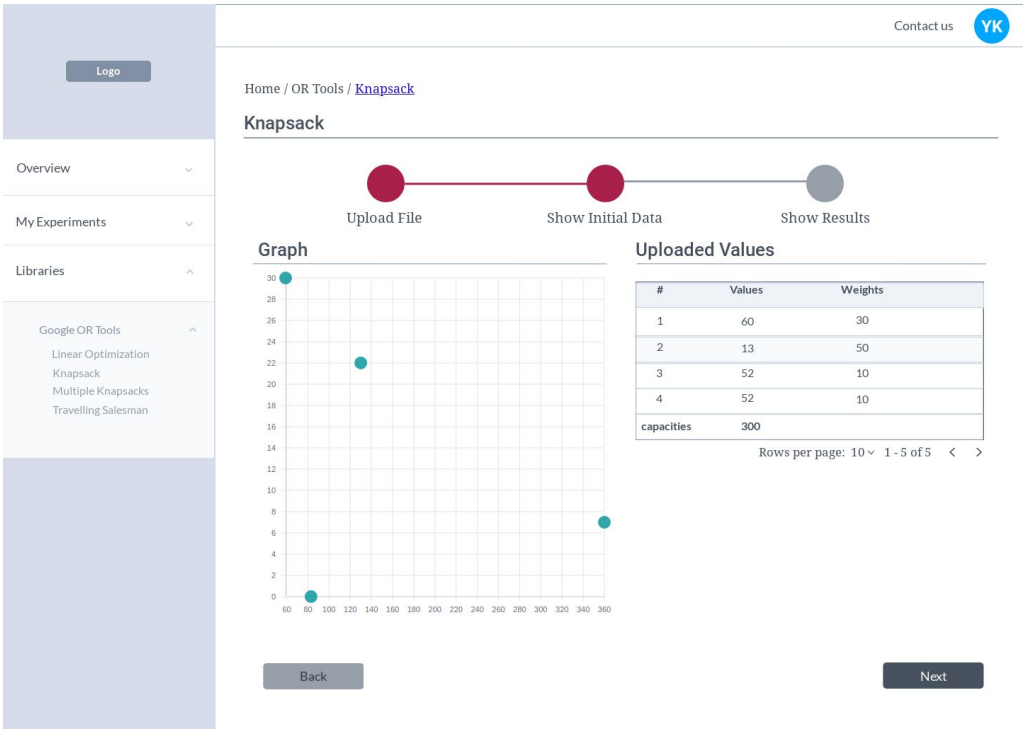
αρχείο για κάθε πρόβλημα. Τέλος εμφανίζονται οι τιμές των πεδίων που διαβάστηκαν από το αρχείο. Τέλος υπάρχει και η δυνατότητα επιλογής κάποιου δείγματος δεδομένων αν ο χρήστης δεν έχει τη δυνατότητα να ανεβάσει αρχείο.



Εικόνα 8: Πρότυπο - Ανέβασμα Αρχείου

3.5.5 Γράφημα με Αρχικά Δεδομένα

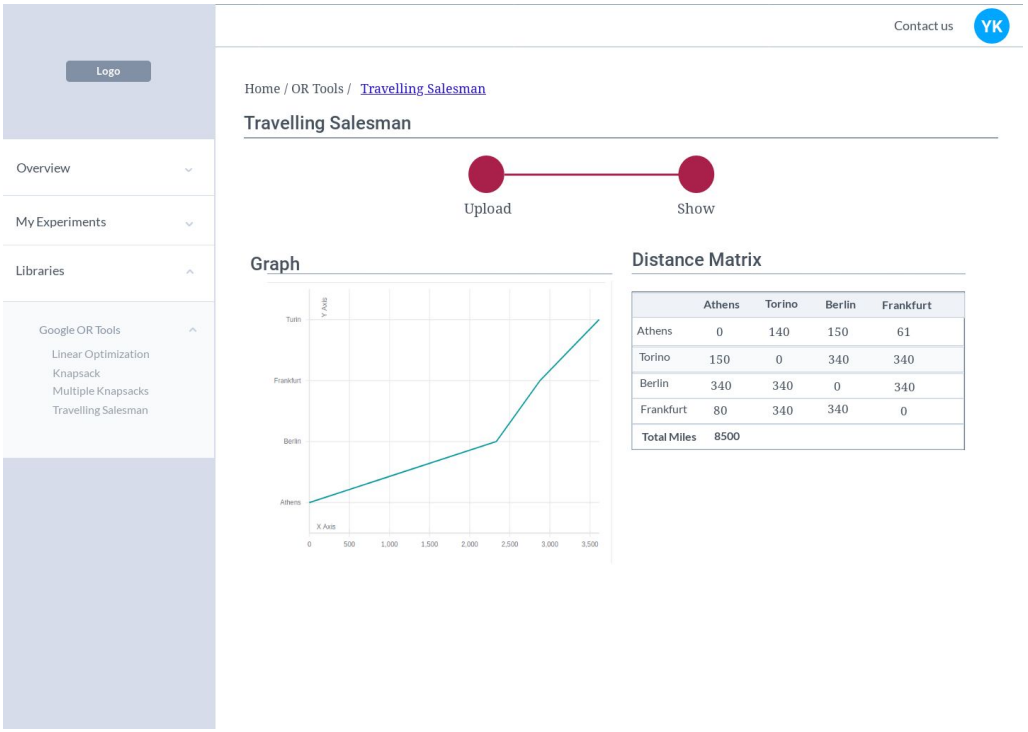
Αρκετά από τα προβλήματα που έχουν μοντελοποιηθεί χρειάζονται την οπτικοποίηση των αρχικών δεδομένων για να συγκριθούν με τα αποτελέσματα. Για αυτό έχει προστεθεί και ένα επιπλέον βήμα πριν από την τελική αναπαράσταση του αποτελέσματος.



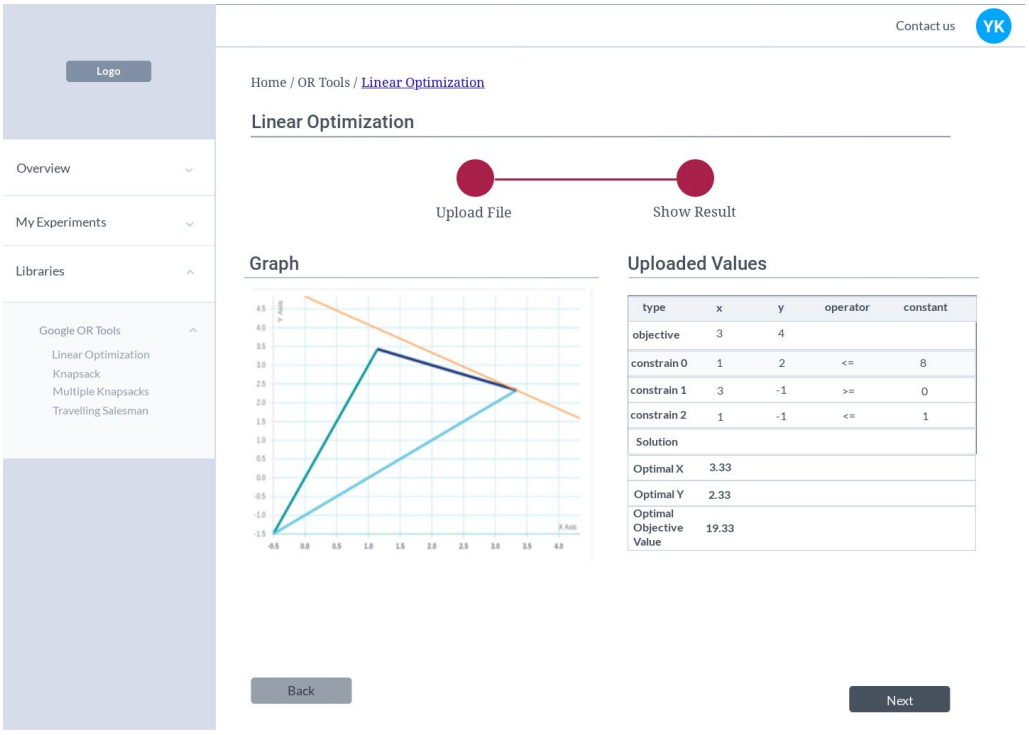
Εικόνα 9: Πρότυπο - Οπτικοποίηση Αρχικών Δεδομένων

3.5.6 Γράφημα για αποτελέσματα

Στο παρακάτω wireframe βλέπουμε την σελίδα με το αποτέλεσμα της εκτέλεσης του εκάστοτε αλγορίθμου. Στην αριστερή πλευρά εμφανίζεται το γράφημα ενώ στην δεξιά εμφανίζονται τα αποτελέσματα σε μορφή πίνακα.



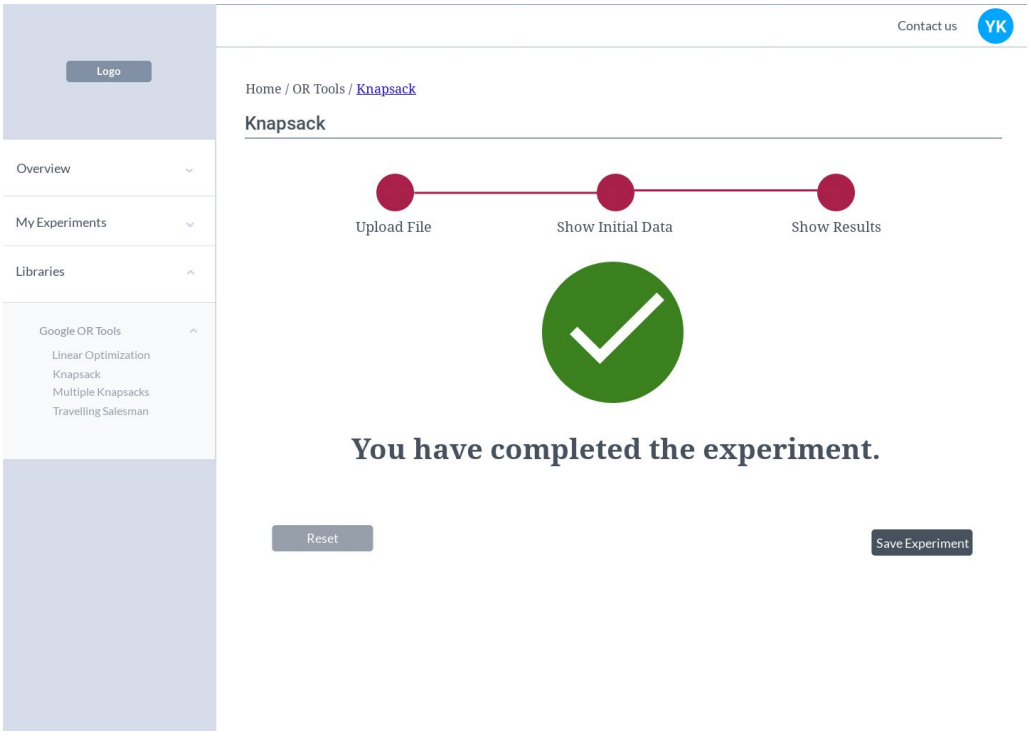
Εικόνα 10: Πρότυπο - Αποτελέσματα για TSP



Εικόνα 11: Πρότυπο - Αποτελέσματα για Γραμμική Βελτιστοποίηση

3.5.7 Σελίδα ολοκλήρωσης πειράματος

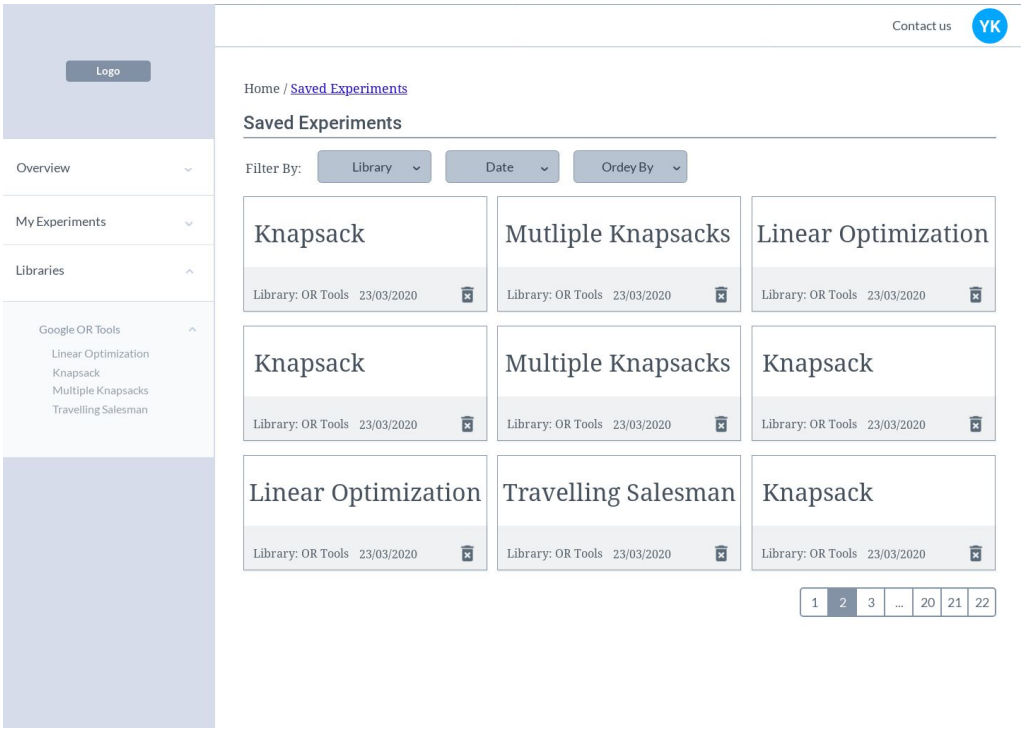
Στη σελίδα αυτή ο χρήστης μπορεί να επιλέξει αν θα αποθηκεύσει το πείραμα που μόλις εκτέλεσε ή όχι καθώς και να κάνει reset και να εισάγει πάλι νέα δεδομένα.



Εικόνα 12: Πρότυπο - Αποθήκευση Πειράματος

3.5.8 Σελίδα ιστορικού πειραμάτων

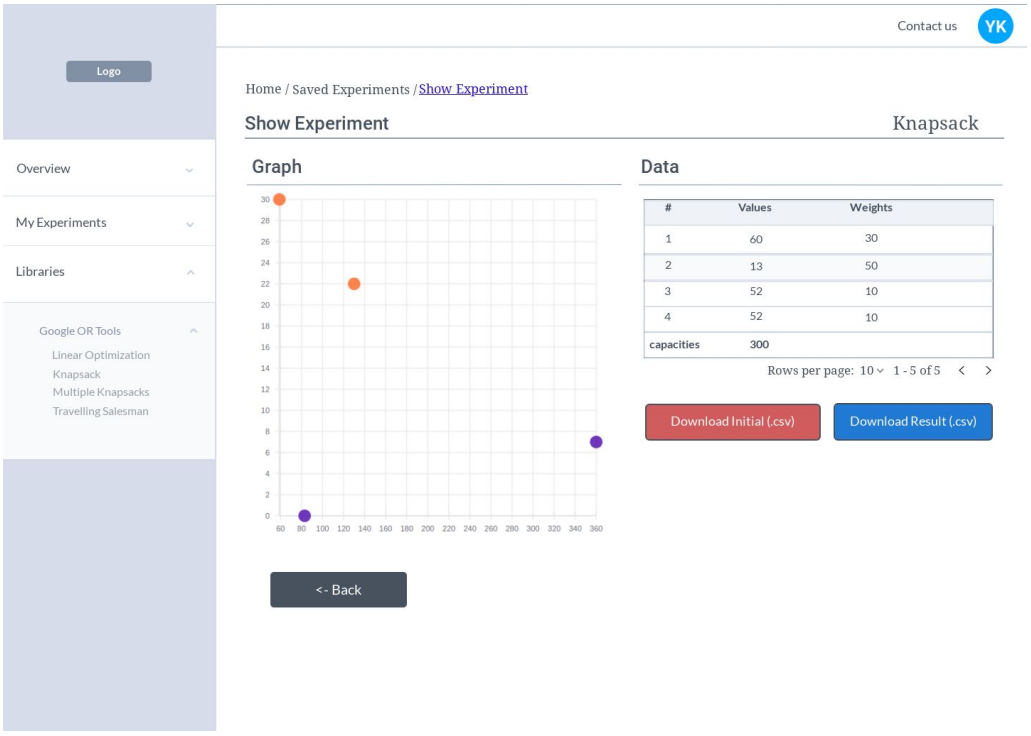
Στο παρακάτω wireframe παρουσιάζεται το ιστορικό των πειραμάτων. Ο χρήστης μπορεί να φιλτράρει με βάση τον αλγόριθμο και να διατάξει τα αντικείμενα με βάση την ημερομηνία εκτέλεσης καθώς και να διαγράψει κάποιο.



Εικόνα 13: Πρότυπο - Αποθηκευμένα Πειράματα

3.5.9 Σελίδα Αποτελέσματος

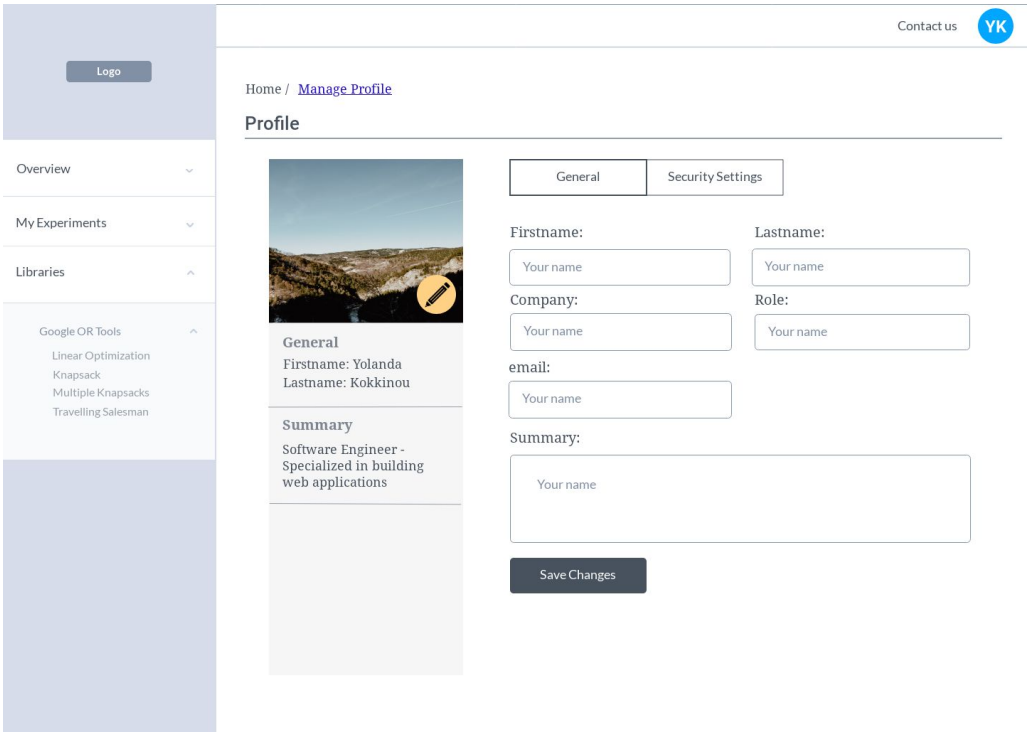
Σε αυτό το πρότυπο ο χρήστης μπορεί να δει ξανά το αποτέλεσμα ενός πειράματος αλλά και να κατεβάσει σε μορφή csv τα αρχικά δεδομένα αλλά και τα αποτελέσματα.



Εικόνα 14: Πρότυπο - Σελίδα Αποτελέσματος

3.5.10 Σελίδα Προφίλ - Αλλαγή Στοιχείων

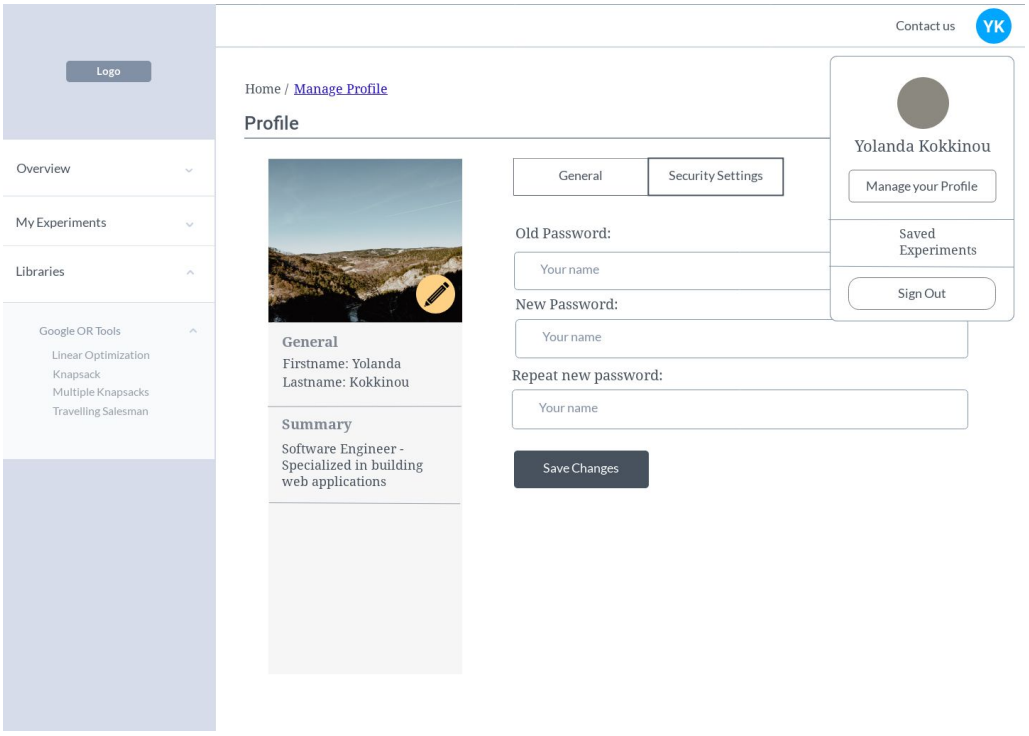
Από το προφίλ μπορεί να γίνει επεξεργασία των στοιχεία του χρήστη όπως είναι το όνομα, το επίθετο του, η εταιρεία στην οποία εργάζεται κ.λ.π.



Εικόνα 15: Πρότυπο - Προφίλ Χρήστη

3.5.11 Σελίδα Προφίλ - Αλλαγή Κωδικού

Επιπλέον δίνεται η δυνατότητα αλλαγής του κωδικού πρόσβασης.



Εικόνα 16: Πρότυπο - Αλλαγή Κωδικού Χρήστη

4. ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

4.1 Νωτιαίο Άκρο Εφαρμογής

Η εφαρμογή είναι υλοποιημένη ως μία μικρουπηρεσία (microservice) και έχει αναπτυχθεί με τη βοήθεια του Spring Boot Framework για το Back End καθώς και του React Framework για το Front End.

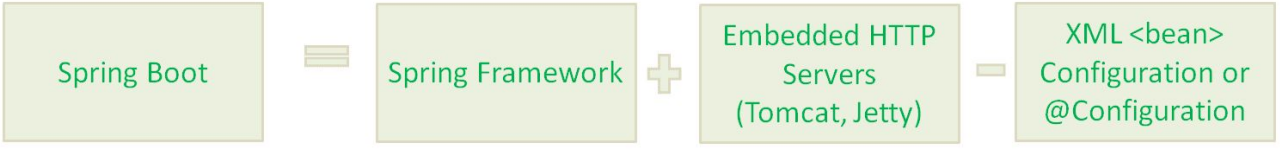
4.1.1 Spring Boot Framework

Το Spring Boot είναι ένα Framework το οποίο κάνει εύκολη την υλοποίηση εφαρμογών διαδικτύου με τη χρήση της γλώσσας JAVA στηριζόμενο στο Spring Framework.

4.1.1.1 Πλεονεκτήματα

Προσφέρει πολλαπλά πλεονεκτήματα σε σχέση με τον κλασικό τρόπο ανάπτυξης web εφαρμογών με τη χρήση του Spring ή άλλων Frameworks για αυτό χρησιμοποιείται και από πολλές επιχειρήσεις:

- **Αυτόματη Παραμετροποίηση (Autoconfiguration):** Το Spring Boot ανάλογα με τα dependencies τα οποία παρατίθενται στο pom.xml μπορεί να παραμετροποιήσει αυτόματα ορισμένες λειτουργίες της εφαρμογής. Για παράδειγμα αν έχουμε ως dependency τη βάση MySQL το spring boot μπορεί να ρυθμίζει αυτόματα τον MySQL connector που θα χρησιμοποιήσουμε.
- **Αυτόνομη (Standalone):** Ίσως το βασικότερο πλεονέκτημα του Spring Boot Framework να είναι ότι δεν χρειάζεται να γίνει deploy κάποιου war αρχείου σε έναν application server όπως ο Tomcat ή ο Wildfly αλλά η εφαρμογή μπορεί να τρέξει ως standalone java application αφού ο Tomcat Server είναι ενσωματωμένος. Αυτό δίνει την δυνατότητα για γρηγορότερο deployment σε περιβάλλον cloud με την λιγότερη δυνατή παραμετροποίηση.
- **Γρηγορότερη Ανάπτυξη (Fast Development):** Συμβάλλει στη γρηγορότερη ανάπτυξη των εφαρμογών και την αποφυγή επαναλαμβανόμενου και τετριμμένου (boilerplate) κώδικα, δηλαδή κώδικα που χρησιμοποιείται για την δημιουργία των CRUD μεθόδων. Το αποτέλεσμα αυτό μπορεί να επιτευχθεί με χρήση διαφόρων annotation του Spring Framework καθώς και με την υλοποίηση διαφόρων interfaces.
- **Περιορισμός των αρχείων XML:** Ένα από τα προβλήματα τα οποία προσπαθεί να λύσει το Spring Boot για τους Java EE (Enterprise Edition) Developers είναι ο περιορισμός των αρχείων XML για την παραμετροποίηση (configuration) της εφαρμογής. Αυτό επιτυγχάνεται με τη χρήση του application.properties αρχείου.



Εικόνα 17: Spring Boot

4.1.1.2 Μειονεκτήματα

Έχει κατηγορηθεί από ορισμένες ομάδες προγραμματιστών για έλλειψη ελέγχου σε περιοχές της εφαρμογής, οι οποίες παραμετροποιούνται αυτόματα από το framework. Επιπλέον αρκετές φορές μπορεί να εγκαθίστανται πολλές εξαρτήσεις (dependencies) τα

οποία δεν χρησιμοποιούνται κατά την υλοποίηση με αποτέλεσμα να αυξάνεται κατά πολύ ο όγκος του αρχείου .jar.

4.1.1.3 Αρχείο application.properties

Μέσω του συγκεκριμένου αρχείου μία εφαρμογή Spring Boot μπορεί να ορίσει οτιδήποτε σχετίζεται με την παραμετροποίηση της web εφαρμογής χωρίς την χρήση XML. Για παράδειγμα:

```
spring.datasource.hikari.connection-timeout=60000
# Max Number of threads to connect with the database
spring.datasource.hikari.maximum-pool-size=6
spring.datasource.hikari.minimum-idle=3
spring.datasource.hikari.idle-timeout=3000
# maximum lifetime in milliseconds of a connection in the pool after it is closed.
spring.datasource.hikari.max-lifetime= 1000
spring.datasource.hikari.poolName=HikariCP
spring.datasource.type=com.zaxxer.hikari.HikariDataSource

# Mysql Local Instance
spring.datasource.url=jdbc:mysql://localhost:3306/ml_algorithms_db?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=password
spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.datasource.hikari.driver-class-name=com.mysql.cj.jdbc.Driver

# Spring Security
security.basic.enabled=false
security.ignored=
jwtSecurityPassword=SecretKeyToGenJWT
# Show hibernate sql for debugging
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Το συγκεκριμένο αρχείο application-local.properties περιέχει την παραμετροποίηση για μία Spring Boot εφαρμογή. Για παράδειγμα οι παράμετροι της μορφής spring.datasource.hikari.* ρυθμίζουν τον αριθμό των thread pools που χρησιμοποιούνται από την εφαρμογή. Στις παραμέτρους της μορφής spring.datasource.* βλέπουμε ότι περιέχονται οι πληροφορίες για σύνδεση με τη βάση ενώ σε αυτές που ακολουθούν το spring.jpa.* μπορούμε να επιλέξουμε τον connector με τη βάση κ.λ.π.

4.1.2 Maven

Το Maven αποτελεί ένα project management tool το οποίο χρησιμοποιείται για να αυτοματοποιήσει την διαδικασία χτισίματος έργων λογισμικού τα οποία χρησιμοποιούν το οικοσύστημα της JAVA.

4.1.2.1 Βασικές αρχές του Maven

Το maven ως εργαλείο ακολουθεί τις παρακάτω αρχές:

- **Αρχεία POM:** Τα pom.xml αρχεία είναι το σημαντικότερο χαρακτηριστικό του maven. Περιέχουν πληροφορίες σχετικές με την παραμετροποίηση του project όπως τις εξαρτήσεις (dependencies), τα πρόσθετα - plugins (π.χ.

spring-boot-maven-plugin) καθώς και τα διάφοροι στόχοι (goals) της εφαρμογής (π.χ. mvn clean, mvn install).

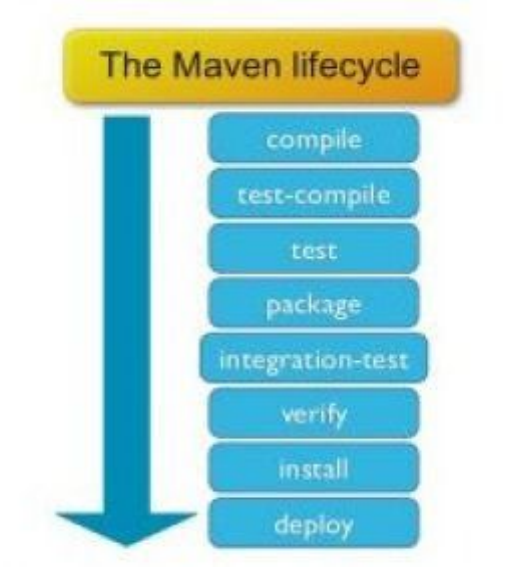
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.2.1.RELEASE</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.or.tools</groupId>
        <artifactId>app</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>app</name>
        <description>Application for OR tools</description>

        <properties>
                <java.version>11</java.version>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-security</artifactId>
                </dependency>
                   <groupId>com.auth0</groupId>
                   <artifactId>java-jwt</artifactId>
                   <version>3.4.0</version>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-jpa</artifactId>
                </dependency>
                <dependency>
                    <groupId>mysql</groupId>
                    <artifactId>mysql-connector-java</artifactId>
            </dependency>
        </dependencies>

        <repositories>
           <repository>
              <id>project.local</id>
              <name>project</name>
              <url>file:${project.basedir}/repo</url>
           </repository>
```

```
        </repositories>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
</project>
```

- **Εξαρτήσεις (Dependencies):** Οι εξαρτήσεις της εφαρμογής είναι ουσιαστικά οι εξωτερικές βιβλιοθήκες της JAVA οι οποίες χρησιμοποιούνται από το έργο λογισμικού και είναι σε μορφή .jar αρχείων. Αρχικά το maven κοιτάει στον τοπικό κατάλογο του συστήματος για να βρεί εάν υπάρχουν τα συγκεκριμένα dependencies στις βιβλιοθήκες αλλιώς τα κατεβάζει μέσω ftp από το κεντρικό εξυπηρέτη του.
- **Κύκλος Λογισμικού Maven και Στόχοι (Maven Lifecycle and Goals):** Ένα ακόμα από τα χαρακτηριστικά του εργαλείου maven είναι ότι μπορούμε να εκτελέσουμε στόχους όπως για παράδειγμα το:

```
mvn clean install
```

Με αυτή την εντολή για παράδειγμα μπορούμε να διαγράψουμε το περιεχόμενο του τοπικού φακέλου και να δημιουργήσουμε αμέσως μετά ένα καινούργιο αρχείο .jar. Πρέπει να τονιστεί ότι εκτελούνται και τα προηγούμενα στάδια σύμφωνα με το παρακάτω διάγραμμα.

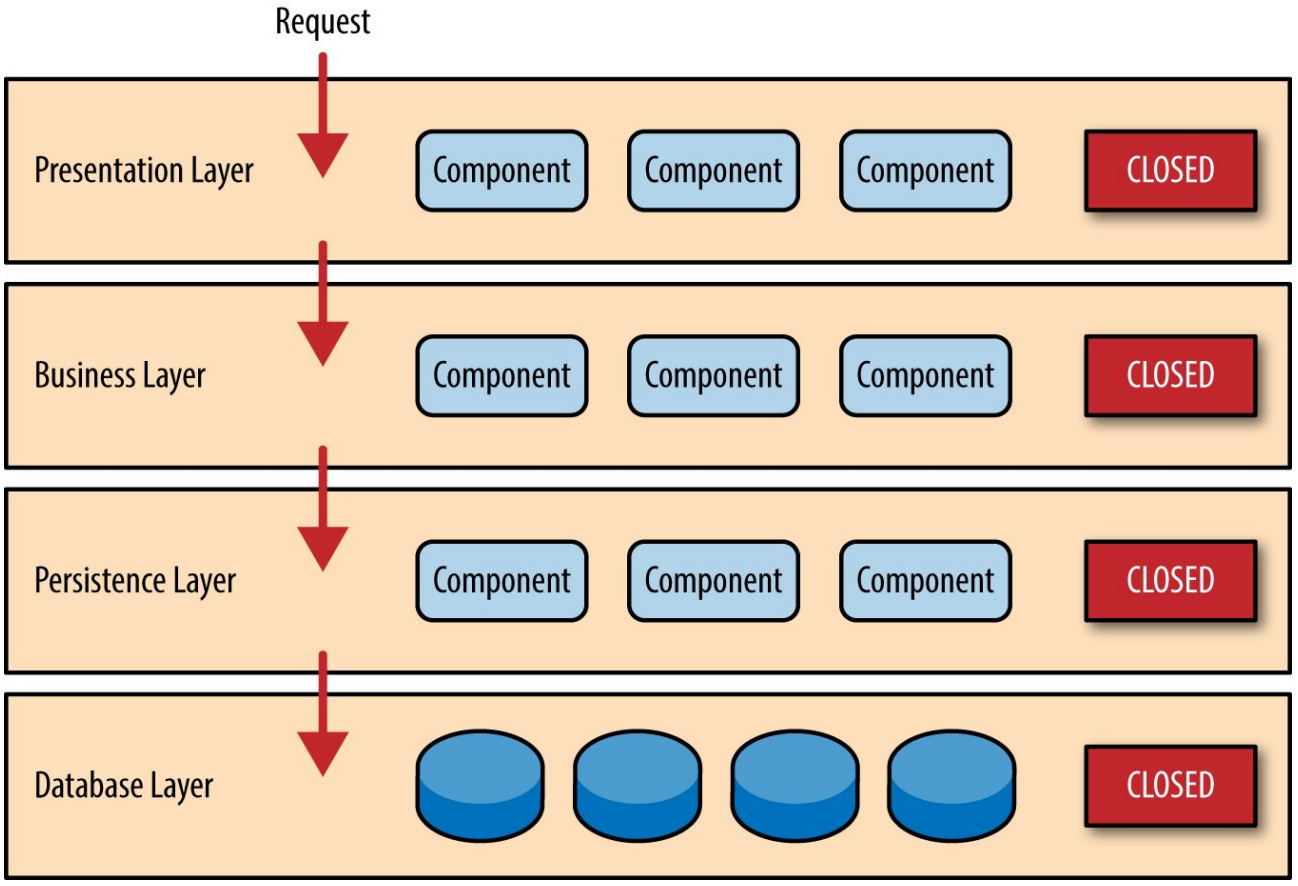


Εικόνα 18: Maven Lifecycle

4.1.3 Πολυστρωματική Αρχιτεκτονική (Multi Layered Architecture)

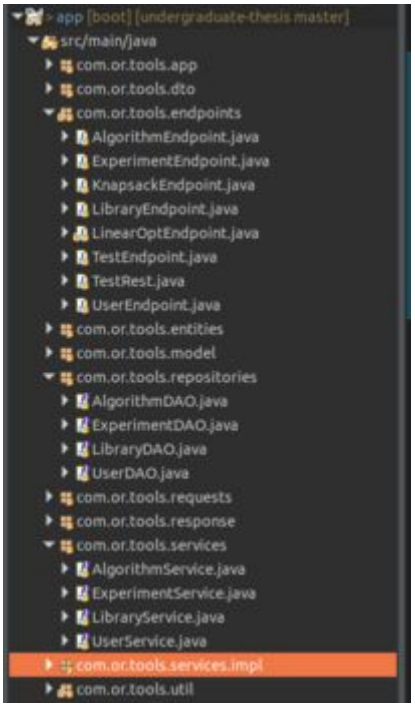
Ένας τρόπος για την αντιμετώπιση της πολυπλοκότητας των σύγχρονων εφαρμογών είναι η υλοποίηση ενός αρχιτεκτονικού μοντέλου που προωθεί την δημιουργία πολλών επιπέδων για τον διαχωρισμό των διαφόρων λειτουργιών της εφαρμογής. Αυτή η αρχιτεκτονική είναι ιδιαίτερα χρήσιμη καθώς οι προγραμματιστές μπορούν να κάνουν

αλλαγές σε κάθε στρώση της εφαρμογής χωρίς να επηρεάσουν κάθε τμήμα της. Οι στρώσεις από τις οποίες αποτελείται είναι οι εξής:



Εικόνα 19: Multi Layered Architecture

- **Στρώση Δεδομένων (Data Layer):** Αυτή η στρώση της εφαρμογής βρίσκεται στο νοητά χαμηλότερο σημείο καθώς χρησιμοποιείται για την αποθήκευση και την ανάκτηση δεδομένων από τη βάση. Στο Spring Framework αυτή η αφαίρεση υλοποιείται από τις κλάσεις που φέρουν το annotation @Repository και συνήθως χαρακτηρίζονται ως αντικείμενα τα οποία είναι υπεύθυνα για την ανάκτηση δεδομένων (Data Access Objects).
- **Στρώση Υπηρεσιών (Service Layer):** Η στρώση της εφαρμογής που αφορά τις υπηρεσίες χρησιμοποιείται για να διαχωρίσει την επιχειρησιακή λογική της εφαρμογής (Business Layer). Το service layer υλοποιείται από το Spring με το annotation @Service το οποίο μπαίνει σε κλάσεις οι οποίες είναι υπεύθυνες για την εκτέλεση μιας επιχειρησιακής λειτουργίας.
- **Στρώση Παρουσίασης (Presentation Layer):** Το κομμάτι αυτό της εφαρμογής υλοποιείται από την στρώση που περιέχει τους διαχειριστές του προτύπου REST, δηλαδή των http endpoints τα οποία χρησιμοποιούνται για την επικοινωνία με τη διεπαφή χρήστη. Το Spring υλοποιεί τον συγκεκριμένο διαχωρισμό με τη βοήθεια του @RestController
  Το σχεδιαστικό αυτό μοντέλο έχει χρησιμοποιηθεί στην εφαρμογή με τους εξής τρόπους:

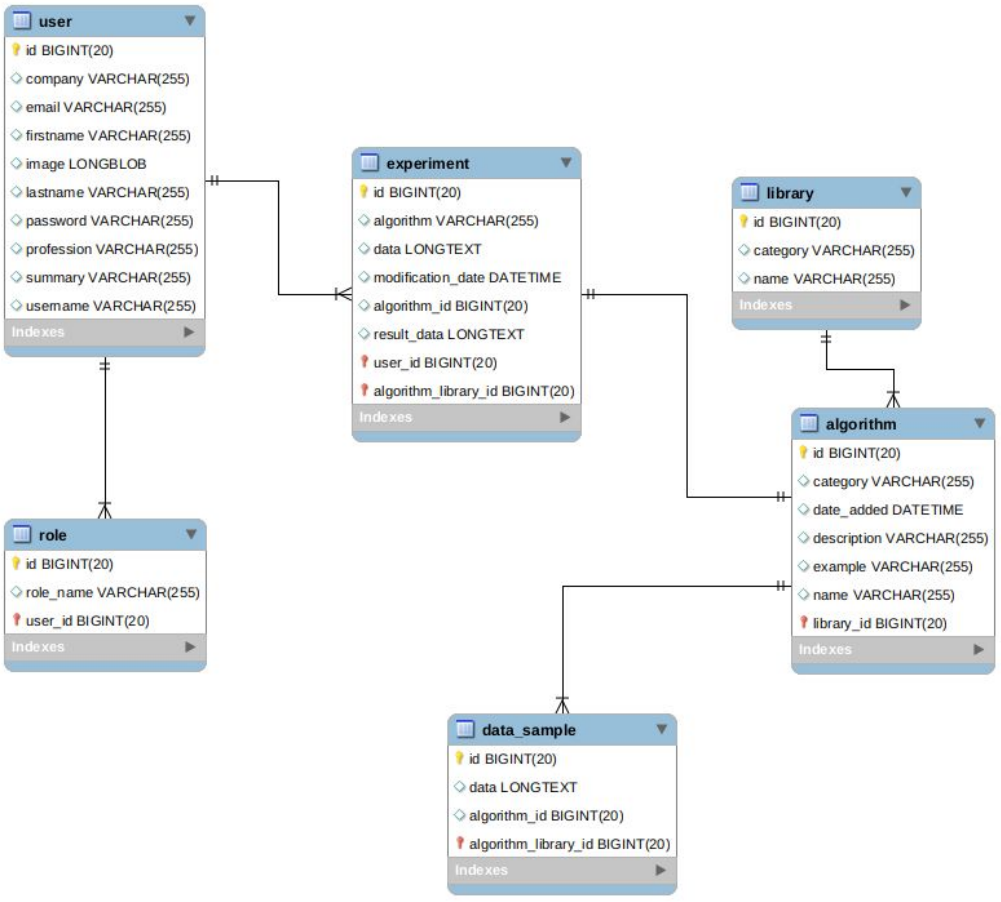


Εικόνα 20: Δομή Κώδικα - Eclipse

Στο πακέτο **com.or.tools.endpoints** βρίσκονται οι κλάσεις οι οποίες σχετίζονται με το **Presentation Layer**, στο πακέτο **com.or.tools.repositories** βρίσκονται οι κλάσεις που σχετίζονται με το **Data Layer** ενώ στο τελευταίο **com.or.tools.service** βρίσκονται οι διεπαφές οι οποίες σχετίζονται με το **Service Layer**.

4.1.4 MySQL

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε η σχεσιακή βάση δεδομένων MySQL η οποία καλύπτει όλες τις ανάγκες για την διαχείριση των δεδομένων.
Το σχήμα της βάσης δεδομένων είναι το εξής:



Εικόνα 21: Σχήμα Βάσης

Οι οντότητες είναι οι παρακάτω:

- **Χρήστης (user):** Περιέχει πληροφορίες για το χρήστη όπως είναι για παράδειγμα, ο κωδικός του κρυπτογραφημένος καθώς και προσωπικές πληροφορίες όπως το όνομα, επίθετο κλπ.
- **Ρόλος (role):** Ο συγκεκριμένος πίνακας περιέχει τους ρόλους του κάθε χρήστη όπως για παράδειγμα admin, user.
- **Πείραμα (experiment):** Ένας χρήστης μπορεί να εκτελέσει και να αποθηκεύσει πολλά πειράματα με βάση κάποιον αλγόριθμο. Αυτά αποθηκεύονται στη βάση και κρατούνται πληροφορίες τόσο για την ημερομηνία εκτέλεσης του πειράματος όσο και για τα δεδομένα που χρησιμοποιήθηκαν σε μορφή JSON.
- **Αλγόριθμος (algorithm):** Περιέχει τους αλγόριθμους που υποστηρίζονται από την εφαρμογή καθώς και τις κατηγορίες τους όπως είναι αλγόριθμοι δρομολόγησης κ.λ.π.
- **Βιβλιοθήκη (library):** Ο συγκεκριμένος πίνακας συνδέεται με τον πίνακα των αλγορίθμων. Με τη βοήθεια αυτού ο χρήστης μπορεί να αναζητήσει τον αλγόριθμο που ψάχνει με βάση την κατηγορία της βιβλιοθήκης.
- **Δείγμα Δεδομένων (data_sample):** Σε αυτόν τον πίνακα αποθηκεύονται διάφορα δείγματα έτσι ώστε ο χρήστης να τα χρησιμοποιήσει αν δεν έχει πρόσβαση σε δεδομένα για την εκτέλεση διαφόρων αλγορίθμων αλλά θέλει να πειραματιστεί.

4.1.5 ORM και JPA

Για την δημιουργία των πινάκων την βάσης χρησιμοποιήθηκε η τεχνική σχεδίασης του Object Relational Mapping (ORM). Με αυτό τον τρόπο ο προγραμματιστής μπορεί να απεικονίσει τους πίνακες που θέλει να δημιουργήσει στη βάση καθώς και τις συσχετίσεις

αυτών μεταξύ τους χωρίς να εγκαταλείψει το Αντικειμενοστραφές Μοντέλο Προγραμματισμού. Στο περιβάλλον της JAVA χρησιμοποιούμε το πρότυπο JPA με το οποίο μέσω διαφόρων annotation μπορούμε να απεικονίσουμε τις οντότητες (entities) τις οποίες θέλουμε να δημιουργήσουμε.
Ένα παράδειγμα μιας κλάσης σε JAVA που αναπαριστά μία οντότητα είναι το εξής:

```
@Entity
@Table(name = "Algorithm")
public class AlgorithmDTO {
        @Id
        @GeneratedValue
        @Column(name = "ID")
        private Long id;
        @Column(name = "NAME")
        private String name;
        @Column(name = "DESCRIPTION")
        private String description;
        @Column(name = "EXAMPLE")
        private String example;
        @Column(name = "DATE_ADDED")
        private Date date;
        @Column(name = "CATEGORY")
        private String category;

        @OneToMany(mappedBy = "algorithm", orphanRemoval = true)
        private List<ExperimentDTO> experiments;

        @ManyToOne(fetch = FetchType.LAZY)
        private LibraryDTO library;

        @OneToMany(mappedBy = "algorithm", orphanRemoval = true)
        private List<DataSampleDTO> dataSamples;
}
```

- **@Entity:** Με το συγκεκριμένο annotation ορίζεται μία οντότητα.
- **@Id:** Με αυτό ορίζουμε ότι το πεδίο της κλάσης που ακολουθεί αποτελεί το κλειδί του αντικειμένου.
- **@GeneratedValue:** Με το συγκεκριμένο annotation ορίζεται η στρατηγική με την οποία το αντικείμενο θα αυτοματοποιήσει τον τρόπο με τον οποίο θα παράγει τα κλειδιά του ο κάθε πίνακας.
- **@OneToMany:** Με το συγκεκριμένο annotation ορίζεται μία σχέση ενός προς πολλά. Όπως στην περίπτωση αυτή ένας αλγόριθμος μπορεί να χρησιμοποιηθεί για την δημιουργία πολλών πειραμάτων (experiments).
- **@ManyToOne:** Με αυτό το annotation ορίζεται μία σχέση πολλά προς ένα όπως για παράδειγμα στη συγκεκριμένη εφαρμογή που πολλά πειράματα μπορούν να εκτελεστούν με τη βοήθεια ενός συγκεκριμένου αλγορίθμου.
- **FetchType:** Με το fetchtype μπορούμε να ορίζεται αν τα δεδομένα των οντοτήτων που συνδέονται με την οντότητα που ανακτάται θα συμπεριληφθούν στο αντικείμενο. Με το eager θα ανακτηθούν ενώ με τον lazy τρόπο ανακτώνται αν και όταν γίνει κάποιο get στη συγκεκριμένη οντότητα. Αυτό το πεδίο σχετίζεται ιδιαίτερα με την απόδοση της εφαρμογής.

4.1.6 Spring Data, JPARepository και JPQL

Τα ερωτήματα (queries) στη βάση δεδομένων από την εφαρμογή γίνονται με τη βοήθεια του Spring Data JPA API και του interface του Spring JPARepository καθώς και με τη βοήθεια της γλώσσας ερωτημάτων JPQL.
Ένα παράδειγμα ενός interface (διεπαφής) που υλοποιεί ένα αποθετήριο (Repository) είναι το παρακάτω:

```
@Repository
public interface ExperimentDAO extends JpaRepository<ExperimentDTO, Long> {
	Page<ExperimentDTO> findByUser(UserDTO user, Pageable page);

	Page<ExperimentDTO> findByUserAndAlgorithm(UserDTO user, AlgorithmDTO algorithm,
Pageable page);

	@Query("SELECT DISTINCT e.algorithm.name FROM ExperimentDTO e WHERE
e.user.username = :username")
	List<String> findAllAlgorithmNames(String username);
}
```

- **JpaRepository Διεπαφή (Interface):** Στο παράδειγμα η κλάση ExperimentDAO επεκτείνει τη διεπαφή JpaRepository. Με την υλοποίηση αυτού του interface το ExperimentDAO κληρονομεί τις βασικότερες μεθόδους που χρειαζόμαστε για τις CRUD λειτουργίες που θέλουμε να εκτελέσουμε στη βάση όπως για παράδειγμα, save, findById και findAll.
- **Spring Δεδομένα (Data):** Με τη βοήθεια του Spring Data μπορούν να δημιουργηθούν συναρτήσεις ακολουθώντας ένα συγκεκριμένο συντακτικό έτσι ώστε να εκτελέσει το Spring ορισμένα named queries. Για παράδειγμα το findByUser μεταφράζεται σε ένα query με το username του χρήστη στο τμήμα του where.
- **JPQL:** Με τη χρήση της συγκεκριμένης γλώσσας στο @Query μπορούν να χρησιμοποιηθούν οι οντότητες που έχουν οριστεί στο μοντέλο έτσι ώστε να επιλεγεί ποιό query θα εκτελεστεί κάθε φορά.

4.1.7 REST και JSON

Τέλος για την επικοινωνία μεταξύ του νωτιαίου και του μετωπιαίου άκρου της εφαρμογής χρησιμοποιείται η αρχιτεκτονική REST και η ανταλλαγή αρχείων JSON. Η αρχιτεκτονική REST χρησιμοποιεί τις κλασσικές κλήσεις HTTP όπως είναι οι POST, GET, PUT, DELETE έτσι ώστε να δημιουργήσει, να διαγράψει ή να ανανεώσει κάποια οντότητα η οποία συνήθως είναι αποθηκευμένη στη βάση. Είναι ιδιαίτερα βοηθητική η χρήση της μετάφρασης των αντικειμένων σε αντικείμενα της Javascript διότι μπορούν άμεσα να χρησιμοποιηθούν από το μετωπιαίου άκρο που είναι γραμμένο σε React.
Με το Spring είναι ιδιαίτερα εύκολο να δημιουργηθούν ορισμένες τερματικές (endpoints) κλάσεις που κάνουν διαθέσιμο ένα API στον χρήστη.
Για παράδειγμα:

```
@RestController
@RequestMapping("/library")
public class LibraryEndpoint {

	@Autowired
	private LibraryService service;
```

```
	@GetMapping("/findAllNames")
	public LibraryNamesResponse getAllLibraryNames() {
		LibraryNamesResponse response = new LibraryNamesResponse();
		response.setLibraryNames(service.getAllLibraryNames());
		return response;
	}
}
```

Η χρήση του annotation @RestController δηλώνει τη δημιουργία ενός Spring Bean και το @RequestMapping ορίζει τη διαδρομή του url. Επιπλέον με το @GetMapping ενεργοποιείται η get λειτουργία των http request που θα δέχεται το συγκεκριμένο endpoint. Αυτό το endpoint μπορεί να ενεργοποιηθεί με την κλήση του παρακάτω URL απο το μετωπιαίο άκρο:

```
curl -H "Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ5b2xhbmRhIiwiZXhwIjoxNTgyMzExNDI0fQ.1
s7YRvZiiXOARf0SN4UpbZOCvuaPWDFuEDhadc4POzbGsQvsEEVAi7r6HF2-bVgOEIftQpuGaBO1JYR31II9IA
" localhost:8080/library/findAllNames
```

4.1.8 Authentication και Authorization με Spring Security

Ένα από τα σημαντικότερα κομμάτια της ανάπτυξης μιας εφαρμογής διαδικτύου είναι η ασφάλεια των δεδομένων των χρηστών, η ταυτοποίηση τους (authentication) και η εξουσιοδότηση (authorization) να εκτελέσουν ορισμένες ενέργειες ανάλογα με το ρόλο τους και τα δικαιώματα (privileges) που έχουν. Αυτό μπορεί να υλοποιηθεί με τη βοήθεια του Spring Security που παρέχει ένα ολοκληρωμένο σύστημα για την ταυτοποίηση και την εξουσιοδότηση των χρηστών.

4.1.8.1 Annotations για Security

Για να οριστεί μία εφαρμογή Spring Boot πρέπει να δημιουργηθεί μία κλάση με το annotation @Configuration όπως η παρακάτω:

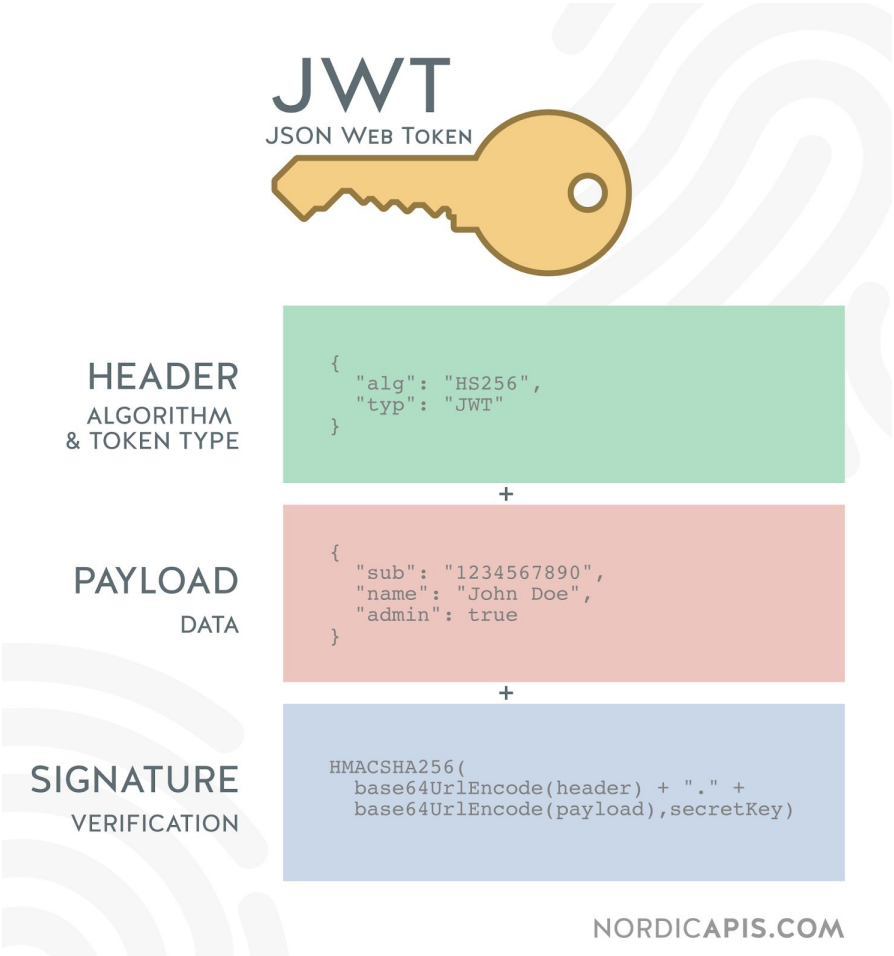
```
@Configuration
@ComponentScan(basePackages = "com.or.tools")
@SpringBootApplication
@EntityScan("com.or.tools.entities")
@EnableJpaRepositories(basePackages = "com.or.tools.repositories")
@EnableTransactionManagement
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class AppApplication {

	public static void main(String[] args) {
		/* Load OR Tools Library */
		System.loadLibrary("jniortools");
		SpringApplication.run(AppApplication.class, args);
	}
}
	@Bean
	public RestTemplate restTemplate(RestTemplateBuilder builder) {
		return builder.build();
	}

	@Bean
```

```
	public BCryptPasswordEncoder passwordEncoder() {
		return new BCryptPasswordEncoder();
	}
}
```

Από αυτή την κλάση για την ασφάλεια χρησιμοποιείται το @EnableSecurity καθώς και το @EnableGlobalMethodSecurity(prePostEnabled = true) στο οποίο γίνεται αναφορά σε επόμενο κεφάλαιο.

4.1.8.2 JWT

Η ταυτοποίηση του κάθε χρήστη γίνεται με το JSON Web Token το οποίο περιέχει ορισμένες πληροφορίες που είναι απαραίτητες για να δοθεί πρόσβαση στο χρήστη σε τερματικά σημεία (endpoints) της εφαρμογής. Το JSON Web Token είναι ένα αντικείμενο της Javascript το οποίο είναι κρυπτογραφημένο με κάποιον κωδικό και δίνεται σε κάθε χρήστη που συνδέεται ή εγγράφεται στην εφαρμογή. Ένα παράδειγμα αυτού είναι το εξής:



Εικόνα 22: JWT - Token

Στο κομμάτι το οποίο αποτελεί τον τίτλο (header) του αντικειμένου υπάρχει ο αλγόριθμος με τον οποίο είναι κρυπτογραφημένα τα δεδομένα, στο payload έχουμε τις πληροφορίες που θέλουμε να αποθηκεύσουμε όπως είναι ο ρόλος του χρήστη στην εφαρμογή (General User, Admin, e.t.c.) και άλλες πληροφορίες όπως το username ή το id του χρήστη. Όλα τα παραπάνω κρυπτογραφούνται με τη βοήθεια ενός κλειδιού το οποίο ορίζεται στο application.properties σε μία Spring Boot εφαρμογή. Με αυτό τον τρόπο διασφαλίζεται ότι

δεν θα δοθεί άδεια τροποποίησης ή ανάκτησης δεδομένων σε μη πιστοποιημένοι χρήστες της εφαρμογής.

4.1.8.3 Φίλτρα σε HTTP Requests

Με το @EnableSecurity και την υλοποίηση ορισμένων interfaces επιτυγχάνεται η ταυτοποίηση του χρήστη για όσα τερματικά σημεία της εφαρμογής είναι αναγκαίο γράφοντας ορισμένους κανόνες. Για παράδειγμα ένας χρήστης πρέπει όταν στέλνει ένα HTTP Request στην εφαρμογή να έχει ένα JSON Web Token στα headers του αιτήματος που στέλνει γιατί αλλιώς θα λάβει το μήνυμα Unauthorized Access 403.
Αρχικά στην κλάση WebSecurity.java ορίζονται όλα τα urls τα οποία απαιτούν την χρήστη του JSON Web Token. Παραδείγματος χάρη το endpoint το οποίο είναι υπεύθυνο για την εγγραφή του χρήστη πρέπει να είναι ελεύθερο και να μην απαιτεί τη χρήση του token.
Κάθε HTTP request το οποίο θα περάσει από την κλάση WebSecurity θα περάσει και μέσα από δύο κλάσεις οι οποίες υλοποιούν κάποια φίλτρα του Spring Security που είναι οι JWTAuthenticationFilter.java και JWTAuthorizationFilter.java με βάση τις οποίες θα ελεχθεί αν το token έχει υπογραφεί από την συγκεκριμένη εφαρμογή και αντίστοιχα θα αποδοθούν ορισμένοι ρόλοι στον χρήστη πράγμα το οποίο αποθηκεύεται στον AuthenticationManager της εφαρμογής που είναι μία ThreadLocal μεταβλητή.

4.1.8.4 hasAuthority και @PreAuthorize

Όπως έχει αναλυθεί και παραπάνω όταν γίνεται το authorization κάποιου χρήστη είναι πιθανό να προστεθεί κάποιος ρόλος ο οποίος να περιορίσει τις δυνατότητες που έχει. Αυτό μπορεί να επιτευχθεί με τον παρακάτω κώδικα πάνω από κάποιο endpoint:

```
@GetMapping(value = "/findAll")
@PreAuthorize("hasAuthority('ROLE_ADMIN')")
public List<ExperimentDTO> findAll() {
	return service.findAll();
}
```

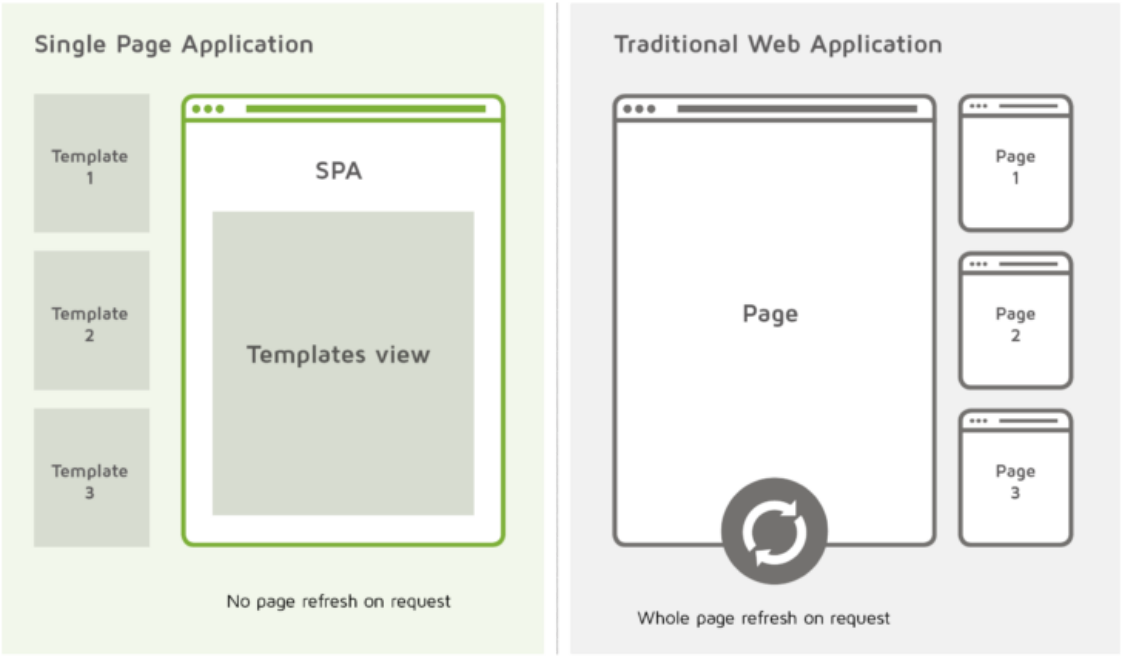
Με αυτό τον τρόπο περιορίζεται η πρόσβαση σε κάποιο endpoint με βάση το ρόλο που έχει ο χρήστης. Στο συγκεκριμένο παράδειγμα επιτυγχάνεται ότι ο απλός χρήστης δεν έχει το δικαίωμα να φέρει από τη βάση όλα τα experiments που έχουν εκτελεστεί και αυτό γίνεται για λόγους απόδοσης.
Φυσικά όμως μπορεί να εφαρμοστούν και κάποιες πιο εξελιγμένες μέθοδοι για την διασφάλιση της πρόσβασης μόνο συγκεκριμένων χρηστών σε κάποιους πόρους του συστήματος. Ένα παράδειγμα είναι το παρακάτω:

```
@PutMapping("/{id}")
@PreAuthorize("@userAuth.checkUser(authentication.name,#id)")
public void userUpdate(@PathVariable Long id, @RequestBody UserResponse request) {
    service.updateUser(id, request.getFirstname(), request.getLastname(),
request.getEmail(), request.getCompany());
}
```

Σε αυτό το endpoint χρησιμοποιείται το @PreAuthorize έτσι ώστε να έχει την δυνατότητα αλλαγής των στοιχείων του μόνο ο συγκεκριμένος χρήστης ο οποίος έχει και το JWT token. Διαφορετικά θα μπορούσε οποιοσδήποτε χρήστης να τροποποιήσει τα δεδομένα του προφίλ ενός άλλου χρήστη.

4.2 Μετωπιαίο Άκρο

Το μετωπιαίο ακρό της εφαρμογής αποτελεί ένα Single Page Application το οποίο δημιουργήθηκε με τη βοήθεια του React Framework καθώς και διαφόρων βιβλιοθηκών όπως η React Vis (οπτικοποίηση των δεδομένων), Material UI (διάφορα έτοιμα αντικείμενα για την διεπαφή χρήστη), Axios (Http Requests για την επικοινωνία με το νωτιαίο άκρο της εφαρμογής).



Εικόνα 23: SPA vs Multi Page

Η συγκεκριμένη αρχιτεκτονική προσφέρει πολλαπλά οφέλη στην εφαρμογή καθώς μία SPA δεν χρειάζονται πολλαπλές επαναφορτώσεις των σελίδων της εφαρμογής και έτσι μειώνεται ο χρόνος αναμονής για την εμφάνιση δεδομένων στον χρήστη. Η επικοινωνία μεταξύ του νωτιαίου και του μετωπιαίου άκρου επιτυγχάνεται με την ανταλλαγή JSON αντικειμένων με http requests.

4.2.1 ReactJS

Το ReactJS αποτελεί ένα σύγχρονο framework για την δημιουργία εφαρμογών διαδικτύου. Υποστηρίζει την τελευταία έκδοση της JavaScript την ECMAScript 6. Το ReactJS αναπτύχθηκε από το Facebook και είναι ένα framework ανοιχτού λογισμικού. Το React στο πρότυπο MVC (Model View Controller) υλοποιεί μόνο το View κομμάτι της εφαρμογής, δηλαδή χρησιμοποιείται για την δημιουργία δυναμικών διεπαφών χρήστη.

4.2.1.1 Χαρακτηριστικά του React

Ορισμένες από τις χαρακτηριστικές ιδιότητες της φιλοσοφίας του React είναι οι εξής:

- **Δηλωτικό (Declarative):** Σχεδιασμός διαφορετικών όψεων για κάθε κατάσταση της εφαρμογής, οι οποίες ανανεώνονται αποτελεσματικά και γρήγορα.
- **Τμήματικό (Component Based):** Δημιουργία τμημάτων (component) τα οποία χρησιμοποιούνται για τον σχεδιασμό πιό περίπλοκων διεπαφών χρήστη.
- **Virtual DOM:** Το React δημιουργεί ένα εικονικό μοντέλο αντικειμένων εγγράφου (Document Object Model) το οποίο διατηρεί μία εσωτερική αναπαράσταση της διεπαφής χρήστη που έχει φορτωθεί. Το συγκεκριμένο μοντέλο ανανεώνεται μόνο όταν έχουμε την αλλαγή ορισμένων δεδομένων.

4.2.1.2 JSX

Μία εφαρμογή του React δομείται με τη βοήθεια αρχείων .jsx τα οποία αποτελούν είτε Functional είτε Class Based Components. Τα αρχεία jsx ουσιαστικά είναι συντακτική ευκολία (syntactic sugar), μοιάζουν με πρότυπα (template) αλλά δεν είναι καθώς μέσα σε αυτά επιτρέπεται να γραφτεί και javascript. Με τη βοήθεια του Babel loader ο οποίος μεταγλωττίζει τα συγκεκριμένα αρχεία σε React αντικείμενα οι προγραμματιστε μπορούν να επωφεληθούν από αυτή την τεχνική.
Παράδειγμα component γραμμένο σε JSX:

```
export default function CustomBreadCrumb(props) {
    let names = props.name.split(',');
    return (
        <Fragment>
            <Breadcrumbs aria-label="breadcrumb" className={styles.breadCrumbStyle}>
                {
                    names.map((text, index)=>(
                        <Link color="inherit" href="/" key={index}>
                            {text}
                        </Link>
                    ))
                }
            </Breadcrumbs>
            <h5>{props.title}</h5>
            <hr className={styles.marginHr}></hr>
        </Fragment>
    ); }
```

Το ίδιο component μεταφρασμένο σε απλή javascript με χρήση των κλάσεων του React:

```
Object.defineProperty(exports, "__esModule", {
  value: true
});
exports.default = CustomBreadCrumb;
function CustomBreadCrumb(props) {
  var names = props.name.split(',');
  return React.createElement(Fragment, null, React.createElement(Breadcrumbs, {
    "aria-label": "breadcrumb",
    className: styles.breadCrumbStyle
  }, names.map(function (text, index) {
    return React.createElement(Link, {
      color: "inherit",
      href: "/",
      key: index
    }, text);
  })), React.createElement("h5", null, props.title), React.createElement("hr", {
    className: styles.marginHr
  }));
}
```

Γίνεται φανερό ότι είναι ευκολότερη η ανάγνωση και η συντήρηση του κώδικα με τη χρήση του JSX.

4.2.1.3 State και Props

Κάθε σύγχρονη εφαρμογή διαδικτύου χρησιμοποιεί και έναν τρόπο για να επεξεργαστεί, να προβάλει και να διαχειριστεί τα δεδομένα. Το React για αυτό το σκοπό χρησιμοποιεί δύο μηχανισμούς: τα props και το state του κάθε Component. Και τα δύο αποτελούν αντικείμενα της javascript αλλά έχουν διαφορετική χρήση.

- **Props:** Τα props χρησιμοποιούνται απο τα React Components και των δύο ειδών και ουσιαστικά λειτουργούν σαν τις παραμέτρους στην κλήση συναρτήσεων. Τα props συνήθως χρησιμοποιούνται κατά την κλίση ενός Child Component και περνάνε σε αυτό ορισμένα δεδομένα από το Parent Component έτσι ώστε να παραμετροποιήσουν το αντικείμενο της διεπαφής του χρήστη. Επιπλέον δεν μπορούν να τροποποιηθούν από το Component το οποίο τα λαμβάνει.

- **State:** Το αντικείμενο state ξεκινά με μία προκαθορισμένη τιμή όταν φορτώνεται από το Component και χρησιμοποιείται για να αποθηκεύει τα δεδομένα τα οποία αλλάζουν λόγω της αλληλεπίδρασης του χρήστη. Δεν χρειάζεται κάθε Component να έχει κατάσταση (state).

4.2.1.4 React Components

Ο κώδικας του React είναι δομημένος σε οντότητες οι οποίες ονομάζονται components. Υπάρχουν δύο διαφορετικά ήδη components τα οποία χρησιμοποιούνται απο το React, τα functional components και τα class based components.

- **Functional Components:** Τα functional components του React ουσιαστικά αποτελούν συναρτήσεις της Javascript. Πολλές φορές χαρακτηρίζονται και ως stateless (χωρίς κατάσταση) διότι απλά δέχονται δεδομένα και τα εμφανίζουν, δηλαδή χρησιμοποιούνται για να απεικονίσουν διάφορα τμήματα της διεπαφής χρήστη.

```
import React, {Fragment} from 'react'
import Alert from "react-bootstrap/Alert";
export const CustomizedAlert = ({value, message, path, componentName}) =>
{
    if(value == null) {
        return (
            ''
        )
    } else {
        return (
            <Alert variant={value}>
                {message}
                <a href={path}>{componentName}</a>
            </Alert>
        )
    }
}
```

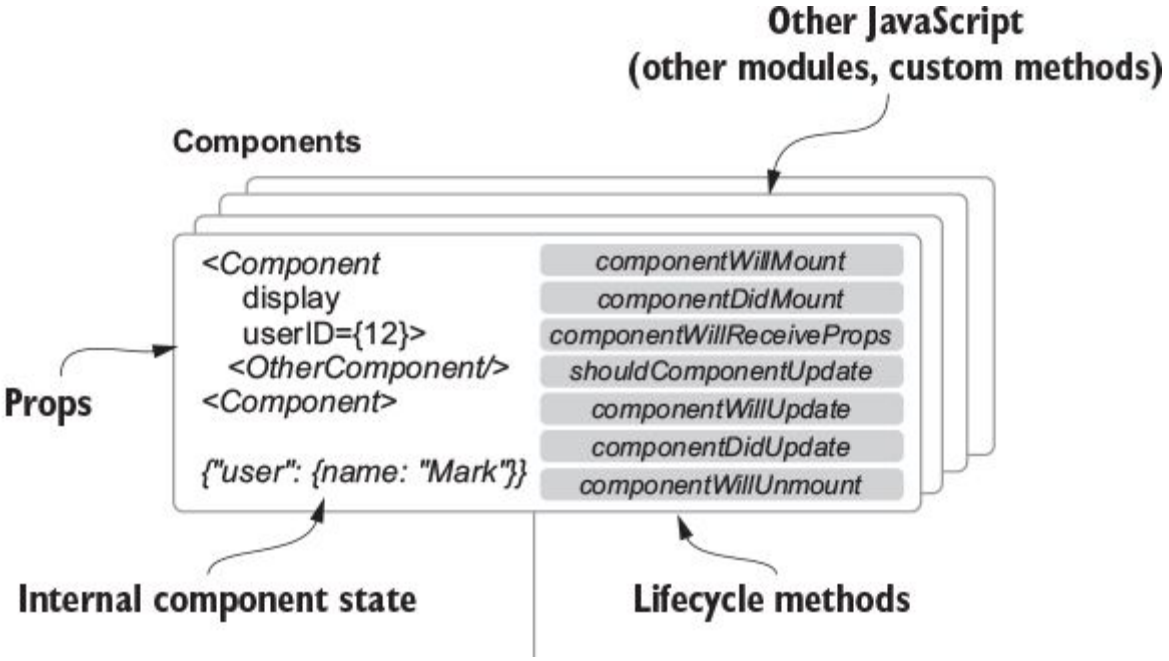
Παραπάνω παρατίθεται ένα παράδειγμα στο οποίο παρουσιάζεται ένα Functional Component το οποίο δέχεται μια σειρά από props παραμέτρους και επιστρέφει ένα alert box. Όπως βλέπουμε ουσιαστικά το συγκεκριμένο είδος Component δεν κρατάει state

αλλά είναι στατικό.

- **Class Based Components:** Τα components (δομικά στοιχεία) τα οποία βασίζονται σε κλάσεις διαφέρουν από τα components τα οποία βασίζονται σε συναρτήσεις. Οι συγκεκριμένες κλάσεις μπορούν να διαχειριστούν props αλλά και state του React ενώ υποστηρίζουν και μερικές ακόμα συναρτήσεις οι οποίες εκτελούνται κατά το Component Lifecycle.
  Για παράδειγμα μπορεί για τις ανάγκες της εφαρμογής να χρειάζεται ορισμένα δεδομένα να είναι διαθέσιμα κατά τη διαδικασία της φόρτωσης του στοιχείου. Τότε είναι πιθανό να προστεθεί αυτό το κομμάτι κώδικα για να εκτελεστεί μέσα σε μία συνάρτηση componentWillMount.

```
export class Homepage extends Component {
    constructor(props) {
        super(props);
        this.state = {
            "firstname": '',
            "lastname": ''
        }
    }

    componentWillMount() {
        axios.get(SERVICE_URL + '/users/' +        localStorage.getItem('username_info'), {
            headers: {"Authorization": localStorage.getItem('authorization')}
        })
        .then((response) => {
                console.log(response);
                this.setState({"firstname": response.data.firstname,
                    "lastname": response.data.lastname});
         },
         (error) => {
              console.log("error");
         });
    }

    componentDidMount() {
        document.body.style.background = "white";
    }

    render() {
        return(
            <div>
                <ResponsiveDrawer firstname={this.state.firstname} lastname={this.state.lastname}/>
                {this.props.children}
            </div>
        );
    }
}

export default withStyles(styles)(Homepage);
```

Στο παραπάνω παράδειγμα παρουσιάζεται ο κώδικα της αρχικής σελίδας της εφαρμογής για τον συνδεδεμένο χρήστη. Για να χρησιμοποιηθούν οι ιδιότητες του React Component είναι απαραίτητο να γίνει extend της κλάση Component ώστε να κληρονομήθούν ορισμένες ιδιότητες που υλοποιεί. Στον κατασκευαστή (constructor) της κλάσης ορίζεται το αντικείμενο του state. Έπειτα αυτό τροποποιείται με την http κλήση στο RESTful API που δημιουργείται όταν το component ετοιμάζεται να φορτωθεί στην διεπαφή χρήστη. Γίνεται δηλαδή χρήση μιας συνάρτησης του κύκλου ζωής του React Component.

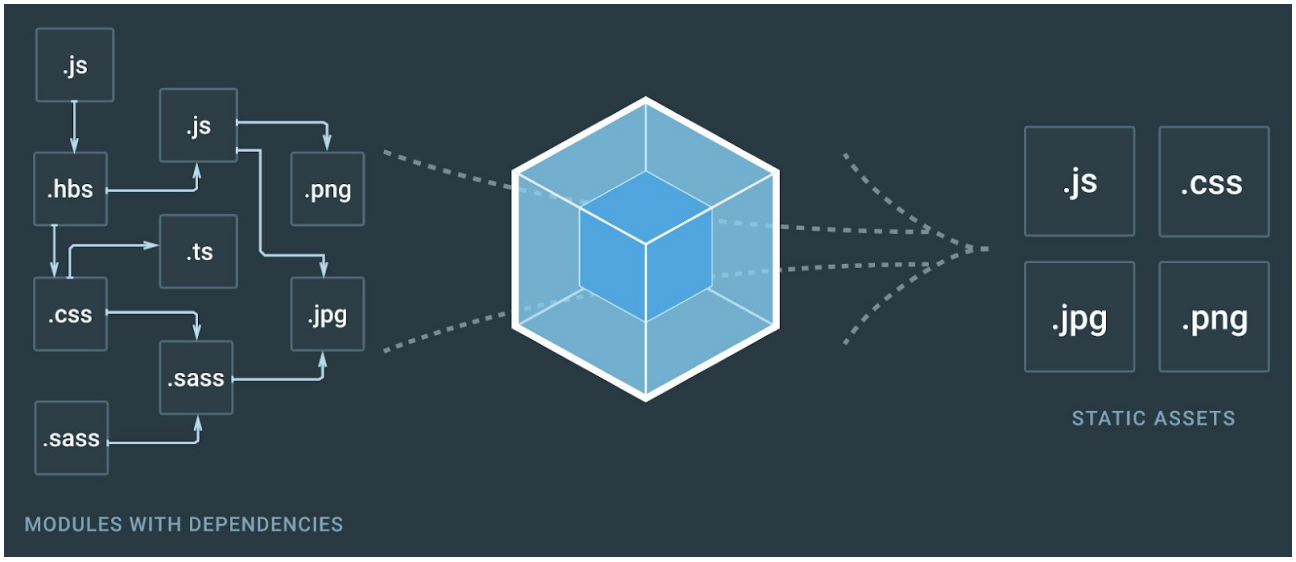


Εικόνα 24: React Lifecycle Methods

4.2.2 Webpack

Στα σύγχρονα javascript frameworks είναι συνηθισμένο η εφαρμογή του μετωπιαίου άκρου να είναι οργανωμένη με τη μορφή modules, δηλαδή διακριτών αρχείων που περιέχουν τμήματα του κώδικα που συνθέτει την λειτουργικότητα της εφαρμογής. Στόχος είναι όμως η δημιουργία ενός ή περισσοτέρων αρχείων javascript τα οποία θα μπορούν να χρησιμοποιηθούν από τον web browser έτσι ώστε να γίνει deploy η εφαρμογή σε έναν εξυπηρετητή. Για αυτό το λόγο είναι απαραίτητη η χρήση ενός εργαλείου όπως είναι το webpack.

4.2.2.1 Πως λειτουργεί το webpack ?



Εικόνα 25: Webpack Transpiling

Το webpack δημιουργεί έναν γράφο εξαρτήσεων (dependency graph) ο οποίος δείχνει τις εξαρτήσεις μεταξύ των modules του κώδικα. Έπειτα ξεκινώντας από κάποια entry points που ορίζονται στο webpack.config.js δημιουργείται ένα αρχείο bundle.js το οποίο μπορεί να φορτωθεί από το φυλλομετρητή ιστού. Είναι σημαντικό αυτό καθώς με αυτό τον τρόπο περιορίζονται τα HTTP Request για τη μεταφορά του αρχείου από το δίκτυο.

4.2.2.2 ES6 και Transpiling

Τα σύγχρονα frameworks της javascript χρησιμοποιούν το πρότυπο ECMASCRIPT 6, στο οποίο έχουν προσθέσει διάφορες νέες ιδιότητες στη γλώσσα. Δυστυχώς όμως αυτό το πρότυπο δεν υποστηρίζεται από όλους τους φυλλομετρητές ιστού. Για αυτό το λόγο έχει προστεθεί στο χτίσιμο μιας web εφαρμογής και η διαδικασία του transpilling. Transpiling καλείται η διαδικασία μεταγλώττισης ενός πηγαίου κώδικα σε μία άλλη μορφή. Για παράδειγμα ο babel είναι ενός τέτοιου είδους transpiler αφού μπορεί να μεταφράσει κώδικα από ES6 σε κώδικα javascript που υποστηρίζεται από τους παλαιότερους browsers. Αυτό μπορεί να επιτευχθεί μέσω των loaders που ορίζονται στο webpack.config.js.

Εικόνα 26: ES6 μετάφραση σε ES5

4.2.2.3 Παράδειγμα αρχείου webpack.config.js

Παρακάτω παρατίθεται το αρχείο webpack.config.js το οποίο χρησιμοποιήθηκε σε κάποιο στάδιο ανάπτυξης της εφαρμογής:

```
const webpack = require('webpack');
const CopyPlugin = require('copy-webpack-plugin');
const path = require('path');

module.exports = {
  entry: path.resolve(__dirname, 'src', 'index.jsx'),
  output: {
    path: path.resolve(__dirname, '../resources/public/output'),
    filename: 'bundle.js',
    publicPath: '/'
  },
  plugins: [
    new CopyPlugin([
      { from: 'src/index.html', to: '../' },
    ]), new webpack.DefinePlugin({
          'SERVICE_URL': JSON.stringify(process.env.productionVar === 'development' ?
            "https://glacial-stream-43144.herokuapp.com" : "http://localhost:8080")
    })
  ],
  resolve: {
    extensions: ['.js', '.jsx', '.min.css', '.css']
  },
  module: {
    rules: [
      {
        test: /\.jsx/,
        use: {
          loader: 'babel-loader',
          options: { presets: ['react', 'es2015'] }
        }
      },
      {
        test: /\.scss/,
        use: ['style-loader', 'css-loader', 'sass-loader']
      },
      {
         test: /\.css$/,
         exclude: /node_modules/,
         use: [{
            loader: "style-loader"
         },
            {
               loader: "css-loader",
               options: {
                  modules: true,
               },
            }
         ],
      }
    ]
  },
  devServer: {
      contentBase: './src',
```

```
    publicPath: '/output',
    port: 9090,
    historyApiFallback: true
  }
};
```

- **Πεδίο εισόδου (entry):** Το σημείο από το οποίο το webpack θα ξεκινήσει να δημιουργεί τον γράφο εξαρτήσεων.
- **Πεδίο εξόδου (output):** Ορίζει το σημείο στο οποίο θα εξαχθεί το bundle.js μετά την διαδικασία του transpilling.
- **Πεδίο φορτωτών (loaders):** Χρησιμοποιούνται για να προεπεξεργαστούν τα διάφορα modules της εφαρμογής.
- **Πεδίο πρόσθετων (Plugins):** Για να γίνει ευκολότερος ο διαχωρισμός μεταξύ των δύο περιβαλλόντων ανάπτυξης που χρησιμοποιούνται μπορούν να περαστούν κάποιες παράμετροι μέσω του webpack όπως για παράδειγμα το SERVICE_URL το οποίο να αλλάζει ανάλογα με το που θα τοποθετηθεί το αρχείο με τον τελικό κώδικα για το μετωπιαίο άκρο bundle.js.
- **Πεδίο κανόνων (rules):** Σε αυτό το σημείο ορίζονται οι κανόνες και τα plugins που χρησιμοποιούνται απο το webpack για να μεταγλωττιστούν ορισμένα αρχεία όπως είναι για παράδειγμα τα .jsx. Αυτά τα αρχεία είναι γραμμένα σε ES6 την οποία όμως δεν αναγνωρίζουν όλοι οι φυλλομετρητές ιστού και έτσι πρέπει μέσω του babel να μεταγλωττιστούν σε μία άλλη μορφή.
- **Πεδίο εξυπηρετητή (devServer):** Όταν αναπτύσσεται μία εφαρμογή που περιλαμβάνει front end κομμάτι μπορεί να σηκωθεί ένας εξυπηρετητής (server) ο οποίος να χρησιμοποιηθεί για την γρηγορότερη ανάπτυξη και την εύρεση λαθών.

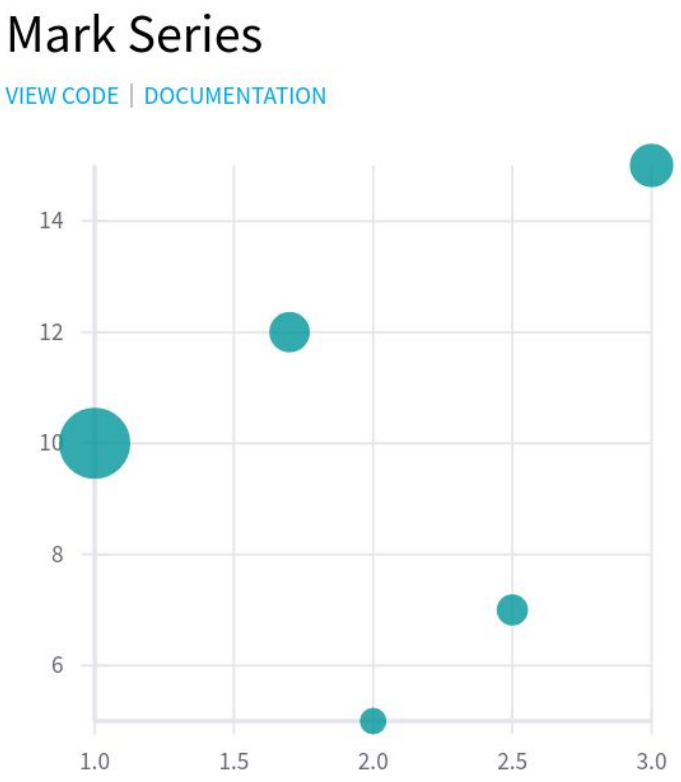
4.2.3 React Vis Library

Όπως είδαμε και παραπάνω η χρήση των σύγχρονων javascript framework είναι εξαιρετικά χρήσιμη γιατί διευκολύνει την επαναχρησιμοποίηση του κώδικα καθώς και τη χρήση διαφόρων βιβλιοθηκών ανοιχτού κώδικα.
Η βιβλιοθήκη React Vis προσφέρει components τα οποία μπορούν να χρησιμοποιηθούν για τη δημιουργία διαφόρων διαγραμμάτων όπως για παράδειγμα είναι τα ραβδογράμματα (bar charts), οι στατιστικές πίτες καθώς και τα διαγράμματα διασποράς (scatter plots).
Ένα παράδειγμα ενός Component που δημιουργεί ένα διάγραμμα διασποράς είναι το παρακάτω:

```
import React from 'react';

import {
  XYPlot,
  XAxis,
  YAxis,
  VerticalGridLines,
  HorizontalGridLines,
  MarkSeries
} from 'index';

export default function Example(props) {
  return (
    <XYPlot width={300} height={300}>
      <VerticalGridLines />
```

```
      <HorizontalGridLines />
      <XAxis />
      <YAxis />
      <MarkSeries
        className="mark-series-example"
        strokeWidth={2}
        opacity="0.8"
        sizeRange={[5, 15]}
        data={[
          {x: 1, y: 10, size: 30},
          {x: 1.7, y: 12, size: 10},
          {x: 2, y: 5, size: 1},
          {x: 3, y: 15, size: 12},
          {x: 2.5, y: 7, size: 4}
        ]}
      />
    </XYPlot>
  );
}
```

Μπορεί να οριστεί με την βοήθεια διαφόρων component ένα γράφημα. Για παράδειγμα γίνεται χρήση του XYPlot για να δημιουργηθούν οι άξονες όπως και τα XAxis και YAxis. Ακόμη ορίζονται διάφορα props τα οποία περνούν ως παράμετροι στα διάφορα components όπως για παράδειγμα το ύψος και το πλάτος του γραφήματος. Τέλος όπως στο MarkSeries component μπορούν να οριστούν και ιδιότητες όπως είναι η διαφάνεια (opacity) του χρώματος του γραφήματος καθώς και το μέγεθος της κάθε κουκίδας.
Με τη βοήθεια του συγκεκριμένου functional component του React δημιουργείται το ακόλουθο διάγραμμα διασποράς.



Εικόνα 27: React Vis - Scatter Plot

4.3 Δημοσίευση (Deployment)

Για την δημοσίευση της εφαρμογής χρησιμοποιήθηκε η πλατφόρμα Heroku η οποία αποτελεί μία πλατφόρμα ως υπηρεσίας (Platform as a Service) στην οποία δημοσιεύονται εφαρμογές εύκολα και γρήγορα.

Εικόνα 28: Heroku Logo

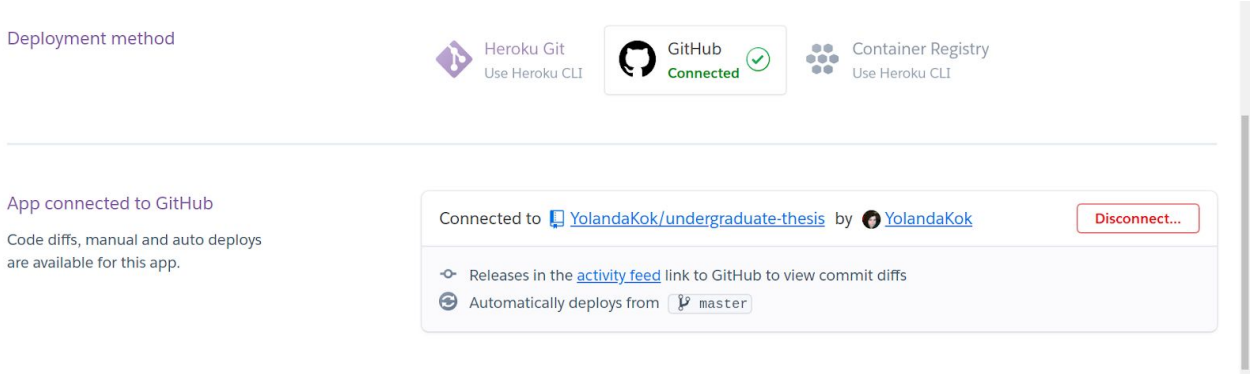
4.3.1 Heroku

Η πλατφόρμα Heroku προσφέρει μία ιδιαίτερα εύκολη διαχείριση τόσο της διαδικασίας της δημοσίευσης της εφαρμογής όσο και της διαχείρισης της. Παρακάτω θα αναλύονται μερικές από τις βασικές ιδιότητες της.

- **Dynos:** Αποτελούν ουσιαστικά τις διεργασίες οι οποίες τρέχουν την εφαρμογή στο cloud. Ανάλογα με τις απαιτήσεις σε πόρους εκάστοτε εφαρμογής μπορεί ο αριθμός αυτών που θα χρησιμοποιηθεί. Για παράδειγμα για το ανέβασμα ενός αρχείου μεγάλου όγκου μπορεί να ενεργοποιηθεί κάποιος worker dyno έτσι ώστε να δημιουργήσει μία άλλη διεργασία που θα αναλάβει το ανέβασμα του αρχείου. Όλα αυτά μπορούν να οριστούν στο Procfile της εφαρμογής.

  ```
  web: java -Dserver.port=$PORT $JAVA_OPTS -Djava.library.path=lib
  -Dspring.profiles.active=dev -jar target/app-0.0.1-SNAPSHOT.jar
  ```

  Για παράδειγμα παραπάνω γίνεται εμφανές ότι με τη βοήθεια ενός web dyno η εφαρμογή μπορεί να εκτελεστεί στο Heroku. Ουσιαστικά χρησιμοποιούνται οι ίδιες παράμετροι που χρησιμοποιούνται και σε μία εφαρμογή Spring Boot για να εκτελεστεί.
- **Continuous Deployment:** Μία από τις βασικότερες λειτουργίες που προσφέρει η συγκεκριμένη πλατφόρμα είναι το συνεχές ανέβασμα του τελευταίου στιγμιότυπου της εφαρμογής μέσω της σύνδεσης του git αποθετηρίου (repository) με το Heroku. Μπορεί να επιλεγεί ένα συγκεκριμένο κλαδί (branch) ενός αποθετηρίου και κάθε φορά που ανεβαίνουν αλλαγές σε αυτό να ανανεώνεται και η εφαρμογή στο Heroku.



Εικόνα 29: Δημοσίευση - Heroku

- **Add-ons (Πρόσθετα):** Ακόμη μία δυνατότητα η οποία βοηθά στην δημοσίευση μιας εφαρμογής είναι και η δωρεάν δοκιμή ορισμένων πρόσθετων, όπως για παράδειγμα είναι η χρήση κάποιας βάσης δεδομένων. Για παράδειγμα χρησιμοποιείται η ClearDB για την δημοσίευση της βάσης στο Cloud.

5. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ - ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

5.1 Γενικά για τη δομή του κώδικα

Ο κώδικας είναι χωρισμένος σε πολλαπλά επίπεδα έτσι ώστε να επιτευχθεί καλύτερη διαχείριση και οργάνωση. Για την μοντελοποίηση των δεδομένων εισόδου και εξόδου του κάθε αλγορίθμου έχουν δημιουργηθεί ορισμένες κλάσεις στο πακέτο com.or.tools.model. Επιπλέον για την επίλυση και την αντιμετώπιση των ιδιαίτερων χαρακτηριστικών κάθε αλγορίθμου έχουν δημιουργηθεί και κάποιες υπηρεσίες (Services) στο πακέτο com.or.tools.algorithms.

5.2 Ανάλυση Αλγορίθμων

Σε αυτό το τμήμα της εργασίας θα αναλυθεί από την πλευρά του νωτιαίου άκρου της εφαρμογής οι μετασχηματισμοί που έγιναν έτσι ώστε να οπτικοποιηθούν τα αποτελέσματα από την πλευρά του μετωπιαίου άκρου για κάθε αλγόριθμο.

5.2.1 Αλγόριθμος Σακίδιου (Knapsack)

Όπως γίνεται αναφορά και στο κεφάλαιο 2 ο αλγόριθμος του σακιδίου έχει ως στόχο την τοποθέτηση μεγίστου αριθμού αντικειμένων σε ένα σακίδιο με περιορισμένη χωρητικότητα. Για την οπτικοποίηση αυτού το αλγορίθμου χρησιμοποιήθηκε ως γράφημα ένα διάγραμμα διασποράς. Για την μοντελοποίηση και επίλυση αυτού του προβλήματος είναι υπεύθυνος ο κώδικας που βρίσκεται στην κλάση KnapsackService.java. Οι συναρτήσεις που την συνθέτουν είναι οι εξής:

- **Pair<List<KnapsackDisplayResponse>, Long> reorganiseData(MultipartFile file)**: Η συγκεκριμένη συνάρτηση δέχεται ως είσοδο ένα αρχείο που έχει σταλεί από τον χρήστη με ένα αίτημα HTTP και επιστρέφει μία λίστα αντικειμένων τύπου KnapsackDisplayResponse καθώς και το μέγεθος του σάκου που είναι τύπου Long. Τα δεδομένα είναι σε τέτοια μορφή έτσι ώστε να εμφανιστούν από το μετωπιαίο (front-end) άκρο της εφαρμογής.
- **KnapsackResult solve(MultipartFile file):** Αυτή η συνάρτηση δέχεται ως είσοδο ένα αρχείο και επιστρέφει το αποτέλεσμα της εκτέλεσης του αλγορίθμου με βάση τον κώδικα που έχει χρησιμοποιηθεί από την βιβλιοθήκη ORTools.

5.2.2 Αλγόριθμος Πολλαπλών Σακιδίων (Multiple Knapsacks)

Ένας ακόμη αλγόριθμος που έχει μοντελοποιηθεί και οπτικοποιηθεί με βάση ένα γράφημα διασποράς είναι ο αλγόριθμος των πολλαπλών σακιδίων. Ο κώδικας για αυτό τον αλγόριθμο βρίσκεται στην κλάση MultipleKnapsacksService.java. Οι συναρτήσεις που την αποτελούν είναι οι εξής:

- **MultipleKnapsackDataRead reorganiseData(MultipartFile file):** Παίρνει ως είσοδο ένα αρχείο το οποίο περιέχει τα βάρη και τις τιμές των αντικειμένων καθώς και το μέγεθος τους κάθε κάδου. Επιστρέφει τα δεδομένα σε μορφή που να μπορούν να οπτικοποιηθούν απο το μετωπιαίο άκρο της εφαρμογής.
- **MultipleKnapsackResponse solve(MultipartFile file):** Παίρνει ως είσοδο ένα αρχείο και με τη βοήθεια του κώδικα των παραδειγμάτων της βιβλιοθήκης ORTools επιλύει το πρόβλημα των πολλαπλών σακιδίων.

5.2.3 Αλγόριθμος Περιπλανώμενου Πωλητή (Travelling Salesman)

Ο αλγόριθμος του περιπλανώμενου πωλητή όπως έχει αναφερθεί ανήκει στην κατηγορία των αλγορίθμων που ψάχνουν να βρουν την συντομότερη διαδρομή δεδομένων ορισμένων πόλεων και των αποστάσεων μεταξύ τους. Για τον υπολογισμού του πίνακα αποστάσεων μεταξύ των πόλεων χρησιμοποιήθηκε το **Distance Matrix**

API της google. Αυτό το API προσφέρει τη δυνατότητα υπολογισμούς των αποστάσεων των πόλεων μέσω HTTP Request. Ένα παράδειγμα αυτού του request είναι το εξής:

```
https://maps.googleapis.com/maps/api/distancematrix/json?units=imperial&origins=Washington,DC&destinations=New+York+City,NY&key=YOUR_API_KEY
```

και η απάντηση είναι:

```
{
   "destination_addresses" : [ "New York, NY, USA" ],
   "origin_addresses" : [ "Washington, DC, USA" ],
   "rows" : [
      {
         "elements" : [
            {
               "distance" : {
                  "text" : "225 mi",
                  "value" : 361715
               },
               "duration" : {
                  "text" : "3 hours 49 mins",
                  "value" : 13725
               },
               "status" : "OK"
            }
         ]
      }
   ],
   "status" : "OK"
}
```

με βάση την οποία βλέπουμε ότι επιστρέφεται στην τιμή value η απόσταση μεταξύ των πόλεων σε μέτρα.

Η επίλυση αυτού του αλγορίθμου υλοποιείται στην κλάση TSPService.java και περιλαμβάνει τις ακόλουθες μεθόδους:

- **long[][] calculateDistanceMatrix(ArrayList<String> cities):** Στη συγκεκριμένη συνάρτηση δημιουργείται ο πίνακας αποστάσεων μεταξύ των πόλεων που περιέχει τις αποστάσεις όλων με όλες.
- **private DistanceMatrixModel send_request(ArrayList<String> origin_addresses, ArrayList<String> dest_addresses):** Δέχεται ως είσοδο δύο λίστες με τις πόλεις που χρησιμοποιούνται ως προορισμοί αλλά και ως σημεία αναχώρησης. Επιστρέφει το αντικείμενο το οποίο χρησιμοποιείται για την οπτικοποίηση των δεδομένων στο μετωπιαίο άκρο της εφαρμογής.

- **String build_address_str(ArrayList<String> addresses):** Δημιουργεί τη μορφή του Request έτσι ώστε να κληθεί το API.
- **RoutesResponse solve(DistanceRequest request):** Δέχεται ως είσοδο τον πίνακα με τις αποστάσεις μεταξύ των πόλεων και το επιλύει με τη βοήθεια των συναρτήσεων της βιβλιοθήκης Google OR Tools. Επιπλέον επιστρέφει μία λίστα η οποία δηλώνει τη βέλτιστη σειρά με την οποία πρέπει ο περιπλανώμενος πωλητής να επισκεφτεί μία πόλη.

5.2.4 Αλγόριθμος Γραμμικής Βελτιστοποίησης (Linear Optimization)

Για τον αλγόριθμο της γραμμικής βελτιστοποίησης γίνεται εισαγωγή των γραμμικών περιορισμών καθώς και της αντικειμενικής συνάρτησης που χρησιμοποιούμε. Για την εύρεση των σημείων τομής μεταξύ των ευθειών χρησιμοποιήθηκε ο αλγόριθμος του Cramer. Η υλοποίηση των συναρτήσεων βρίσκεται στην κλάση LinearOptService.java και είναι η εξής:

- **Pair<List<List<Double>>, Double> result(LinearOptModel model):** Η συνάρτηση αυτή δέχεται ως είσοδο τους γραμμικούς περιορισμούς καθώς και την αντικειμενική συνάρτηση. Επιστρέφει το σημείο τομής των περιορισμών με την αντικειμενική συνάρτηση καθώς και άλλα δύο σημεία για να οριστεί καλύτερη αυτή η ευθεία και να οπτικοποιηθεί.
- **List<List<Double>> morePoints(Double x, Double y, Double constant, LinearObjective obj):** Βρίσκει δύο τυχαία σημεία για την αντικειμενική συνάρτηση έτσι ώστε να αναπαρασταθεί.
- **List<List<Double>> solveCramer(LinearOptModel model):** Με τον αλγόριθμο του Cramer βρίσκει τα σημεία τομής μεταξύ των περιορισμών.

6. ΠΑΡΟΥΣΙΑΣΗ ΕΦΑΡΜΟΓΗΣ

6.1 Τελικό Αποτέλεσμα

Στο παρακάτω κεφάλαιο θα παρουσιαστεί το τελικό αποτέλεσμα της εφαρμογής καθώς και ορισμένα παραδείγματα χρήσης της. Όπως έχουμε αναφέρει και παραπάνω η εφαρμογή έχει την δυνατότητα ανεβάσματος αρχείων με δεδομένα για την επίλυση συγκεκριμένων προβλημάτων που αφορούν την επιχειρησιακή έρευνα καθώς και γνωστά αλγοριθμικά προβλήματα με τα οποία ασχολείται η επιστήμη των υπολογιστών. Έχει γίνει ιδιαίτερη προσπάθεια έτσι ώστε να δημιουργηθεί μία σύγχρονη και εύχρηστη εφαρμογή διαδικτύου για την εμφάνιση και αποθήκευση αποτελεσμάτων.

6.2 Μελλοντικές Επεκτάσεις

Για τον σχεδιασμό και την υλοποίηση της συγκεκριμένης εφαρμογής έχουν χρησιμοποιηθεί σύγχρονα frameworks και βιβλιοθήκες λογισμικού όπως είναι το Spring Boot, το React js και επιμέρους βιβλιοθήκες του όπως είναι το React vis της εταιρείας Uber το οποίο ασχολείται με την οπτικοποίηση δεδομένων. Με αυτό τον τρόπο προσπάθησα να διασφαλίσω ότι ο κώδικας θα είναι εύκολα επεκτάσιμος από όποιον θέλει να προσθέσει αργότερα και άλλα αλγοριθμικά προβλήματα.
Στη συγκεκριμένη έκδοση του λογισμικού παρουσιάζονται τα προβλήματα Knapsack (Σακιδίου), Multiple Knapsacks (Πολλαπλών Σακιδίων), Travelling Salesman (Περιπλανώμενου Πωλητή) καθώς και Linear Optimization για επίλυση γραμμικών προβλημάτων περιορισμών με 2 μεταβλητές (Γραμμική Βελτιστοποίηση).
Φυσικά σε μελλοντικές εκδόσεις θα μπορούσαν να αξιοποιηθούν και άλλα προβλήματα από τις κατηγορίες του Routing (Δρομολόγησης) και Packing (Ομαδοποίησης) που επιλύει η βιβλιοθήκη Google OR Tools.
Η διαδικασία που θα πρέπει να ακολουθηθεί είναι:

1. Εντοπισμός των δεδομένων εισόδου
2. Μοντελοποίηση δεδομένων του αρχείου εισόδου
3. Δημιουργία parser στο backend
4. Μετασχηματισμός Δεδομένων εισόδου
5. Αποστολή τροποποιημένων δεδομένων εισόδου στο νωτιαίο άκρο
6. Οπτικοποίηση των αρχικών δεδομένων
7. Δημιουργία endpoint και service κλάσης που χρησιμοποιεί τον κώδικα του Google OR tools για την επίλυση των δεδομένων
8. Αποστολή του τελικού αποτελέσματος στο μετωπιαίο άκρο για οπτικοποίηση από οποιαδήποτε βιβλιοθήκη οπτικοποίησης έχει επιλεχθεί

Με βάση αυτή τη διαδικασία ο προγραμματιστής μπορεί να προσθέσει και άλλα πειράματα τα οποία θα μπορεί να εκτελέσει ο χρήστης.

6.3 Παρουσίαση Εφαρμογής

Σε αυτό το τμήμα της παρούσας πτυχιακής εργασίας γίνεται παρουσίαση του τελικού αποτελέσματος και δίνονται παραδείγματα εκτέλεσης των αλγορίθμων που οπτικοποιούνται. Επιπλέον δίνονται και μερικά πρότυπα αρχείων για την εισαγωγή των αρχικών δεδομένων.

6.3.1 Αρχική Σελίδα

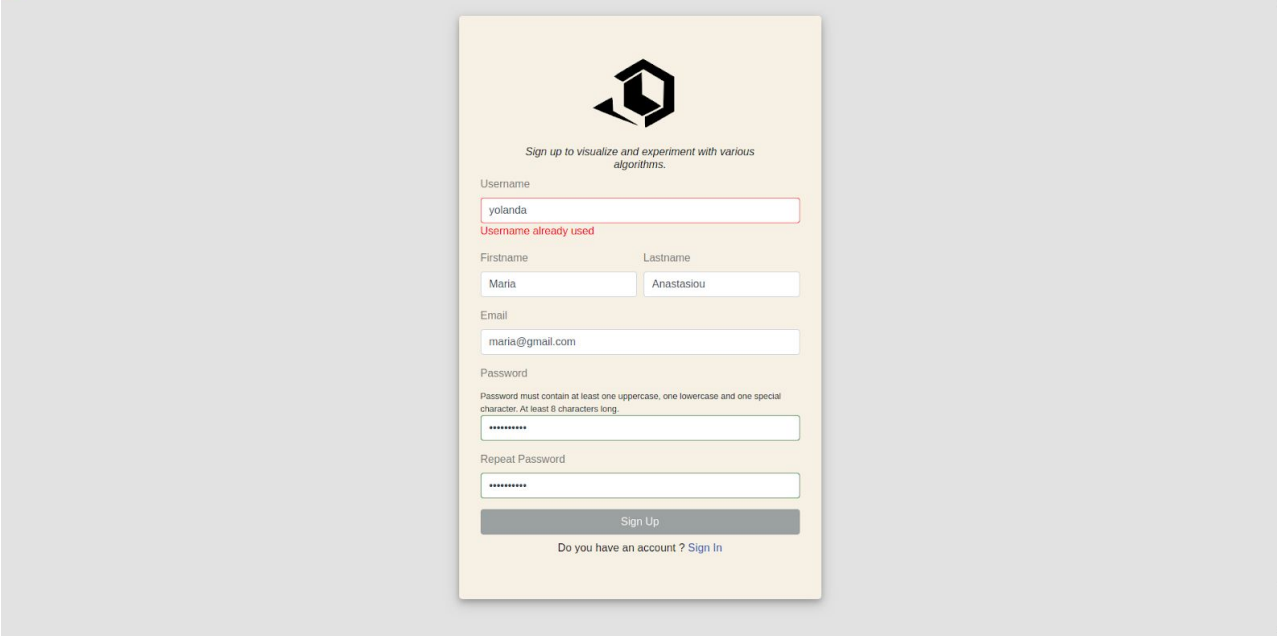
Παρακάτω φαίνεται το τελικό αποτέλεσμα της αρχικής σελίδας όταν ο χρήστης μεταβεί στην αρχική σελίδα της εφαρμογής Visualize Zone. Ο χρήστης μπορεί να επιλέξει αν θα

συνδεθεί αν έχει ήδη λογαριασμός ή θα δημιουργήσει. Πάνω αριστερά παρουσιάζεται το όνομα της εφαρμογής ενώ δεξιά βρίσκονται τα sign in και sign up κουμπιά της εφαρμογής.

Εικόνα 30: Αρχική Σελίδα

6.3.2 Σελίδα Εγγραφής

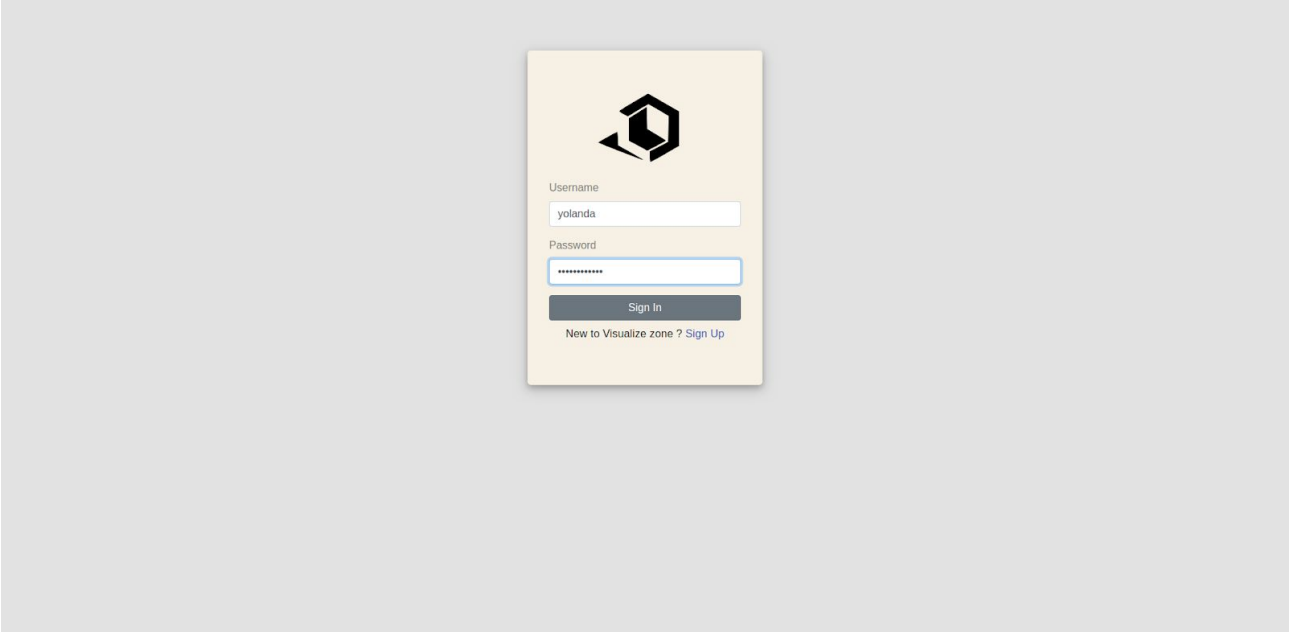
Στην παρακάτω εικόνα βλέπουμε την σελίδα εγγραφής του χρήστη. Για το σχεδιασμό της έχει ακολουθηθεί η μεθοδολογία του Nielsen για την ειδοποίηση του χρήστη για τον αν έχει εισάγει το σωστό pattern για τον κωδικό ή για το αν το όνομα χρήστη που έχει επιλέξει χρησιμοποιείται ήδη από κάποιον άλλο χρήστη. Αν ο χρήστης συμπληρώσει σωστά τα δεδομένα ενεργοποιείται το κουμπί για να υποβάλλει τη φόρμα.



Εικόνα 31: Εγγραφή

6.3.3 Σελίδα Εισόδου

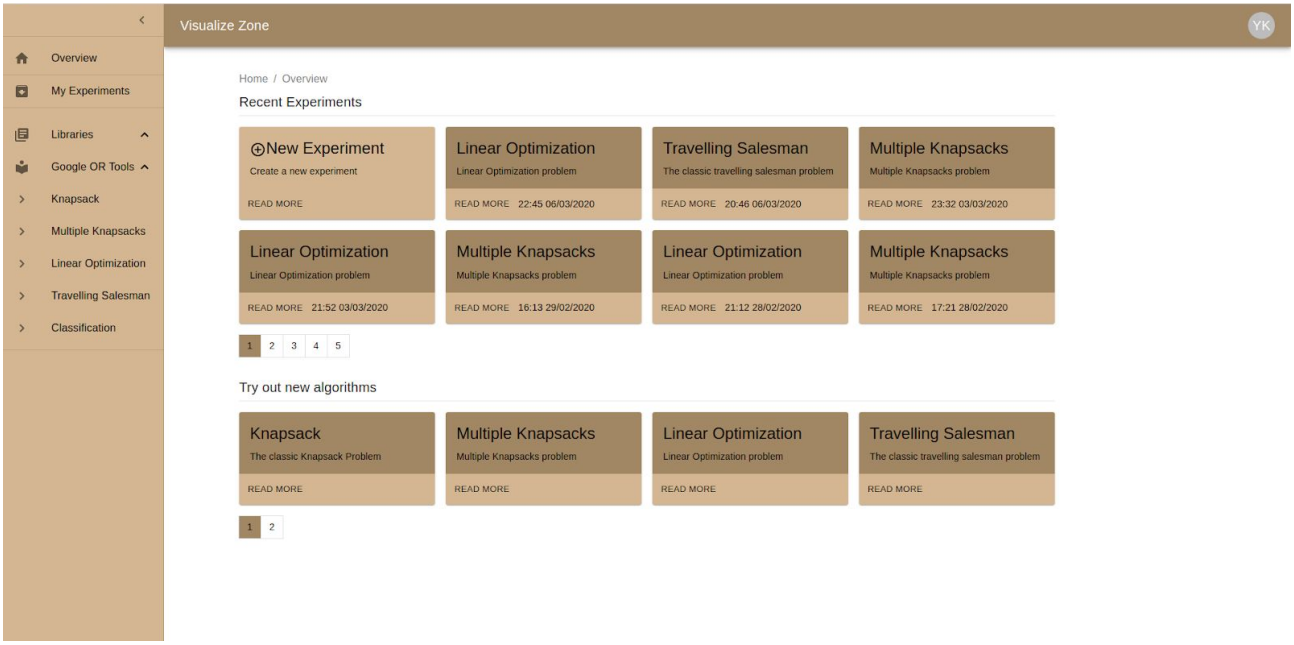
Σε αυτή την εικόνα βλέπουμε την σελίδα εισόδου της εφαρμογής. Ο χρήστης πρέπει να εισάγει το username και το password του. Αν δεν έχει λογαριασμό μπορεί να πατήσει στο Sign Up και να δημιουργήσει.



Εικόνα 32: Σύνδεση Χρήστη

6.3.4 Κεντρική Σελίδα (Overview)

Στην κεντρική σελίδα μετά τη σύνδεση ο χρήστης μπορεί να δει μία σελιδοποιημένη λίστα των πρόσφατων αποτελεσμάτων που έχει εκτέλεση. Επιπλέον μπορεί να επιλέξει την κάρτα που λέει “New Experiment” έτσι ώστε να δημιουργήσει ένα νέο πείραμα με βάση τους αλγορίθμους που προσφέρονται από την εφαρμογή. Στο κάτω μέρος μπορεί να δοκιμάσει απευθείας τους πρόσφατους αλγορίθμους που έχουν προστεθεί από τους διαχειριστές της εφαρμογής. Στο αριστερό μέρος μπορεί να μεταβεί μέσω του sidebar σε όλα τα πειράματα που έχει δημιουργήσει “My Experiments” και να τα αναζητήσει ή να διαγράψει ορισμένα. Στο κάτω τμήμα του sidebar βλέπουμε τους αλγορίθμους ανα βιβλιοθήκη που προσφέρονται.

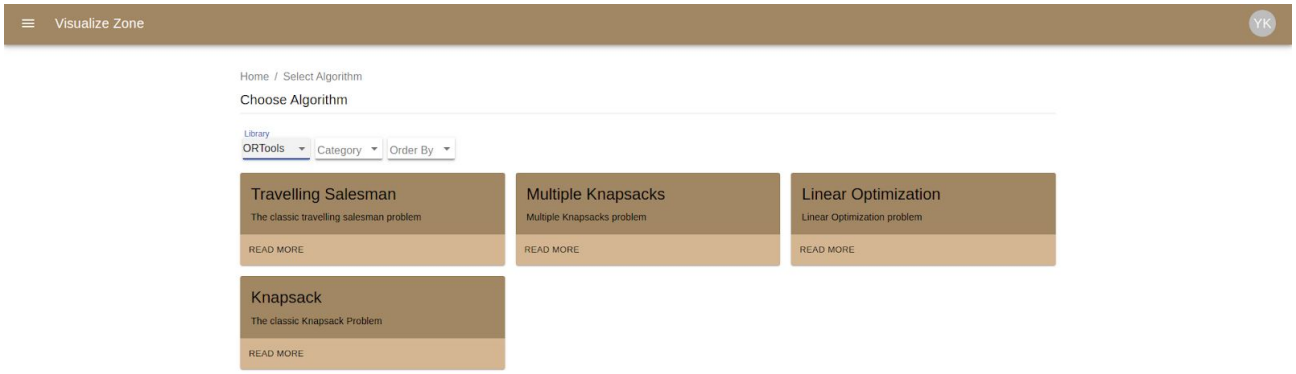


Εικόνα 33: Κεντρική Σελίδα Εφαρμογής - Συνδεδεμένος Χρήστης

6.3.5 Σελίδα Επιλογής Αλγορίθμου

Αυτό το κομμάτι της εφαρμογής βλέπει ο χρήστης όταν πατήσει πάνω στην κάρτα “New Experiment” της αρχικής σελίδας. Από εδώ μπορεί να επιλέξει ποιόν αλγόριθμο θέλει να

εκτελέσει καθώς και να φιλτράρει τα αποτελέσματα ανάλογα με την βιβλιοθήκη ή την κατηγορία του αλγορίθμου (Packing, Routing, e.t.c.) ή να τα διατάξει αλφαβητικά.



Εικόνα 34: Επιλογή Αλγορίθμου

6.3.6 Αλγόριθμος Σακιδίου (Knapsack)

Σε αυτή την ενότητα παρουσιάζεται η διαδικασία για την οπτικοποίηση του αλγορίθμου του σακιδίου καθώς και η αποθήκευση των δεδομένων από το χρήστη.

6.3.6.1 Αρχείο για Αλγόριθμο Σακιδίου (Knapsack)

Το αρχείο το οποίο θα εισαχθεί θα πρέπει να περιλαμβάνει δύο στήλες, μία με τις τιμές (values) και μία με τα βάρη (weights) των αντικειμένων. Επιπλέον πρέπει στην τελευταία του γραμμή να υπάρχει το capacity δηλαδή το όριο του βάρους που θέλουμε να έχει ο σάκος μας. Ακόμη τα δεδομένα στο αρχείο πρέπει να χωρίζονται με το σύμβολο (;) και το αρχείο να είναι σε μορφή (.csv).
Ένα παράδειγμα είναι το εξής:

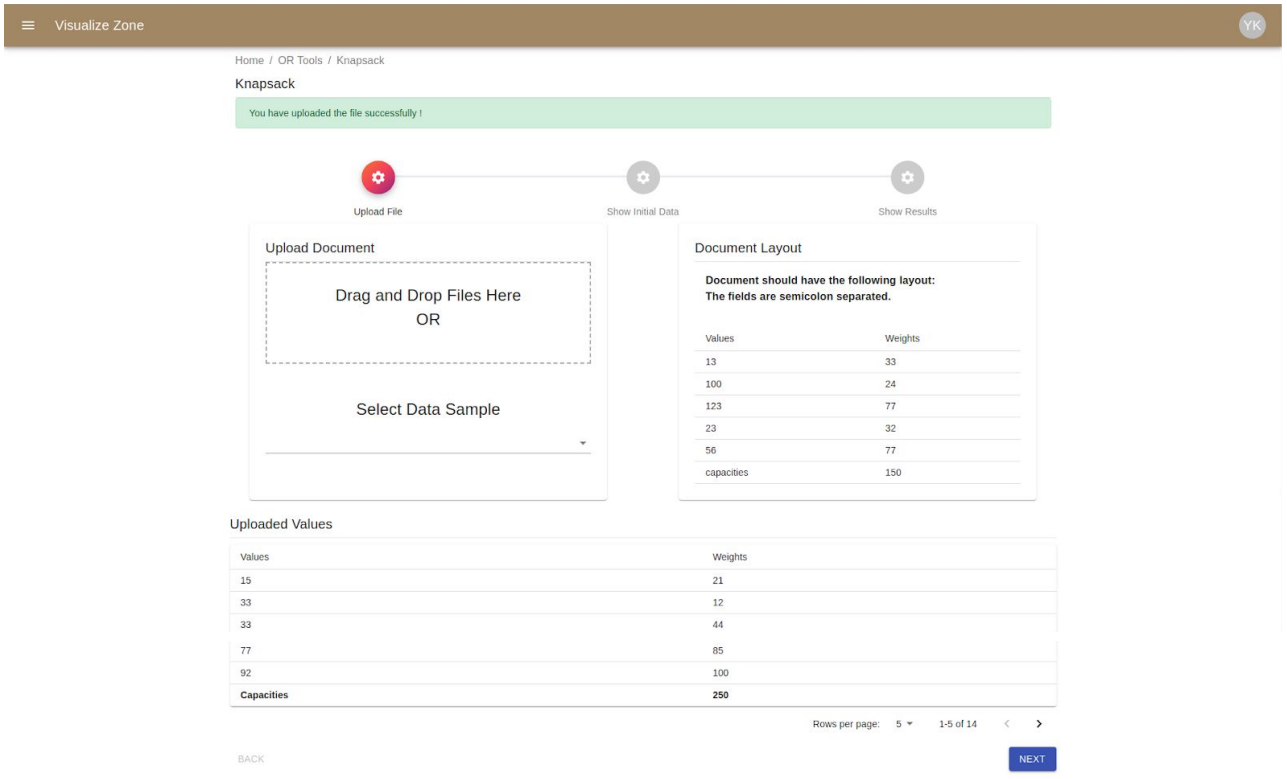
Πίνακας 1: Δεδομένα Εισόδου (Knapsack)

values	weights
15	21
33	12
33	44
77	85
92	100
120	35
33	200
89	52
27	27

32	56
78	99
13	45
34	56
300	120
capacities	250

6.3.6.2 Ανέβασμα Αρχείου - Αρχικά Δεδομένα

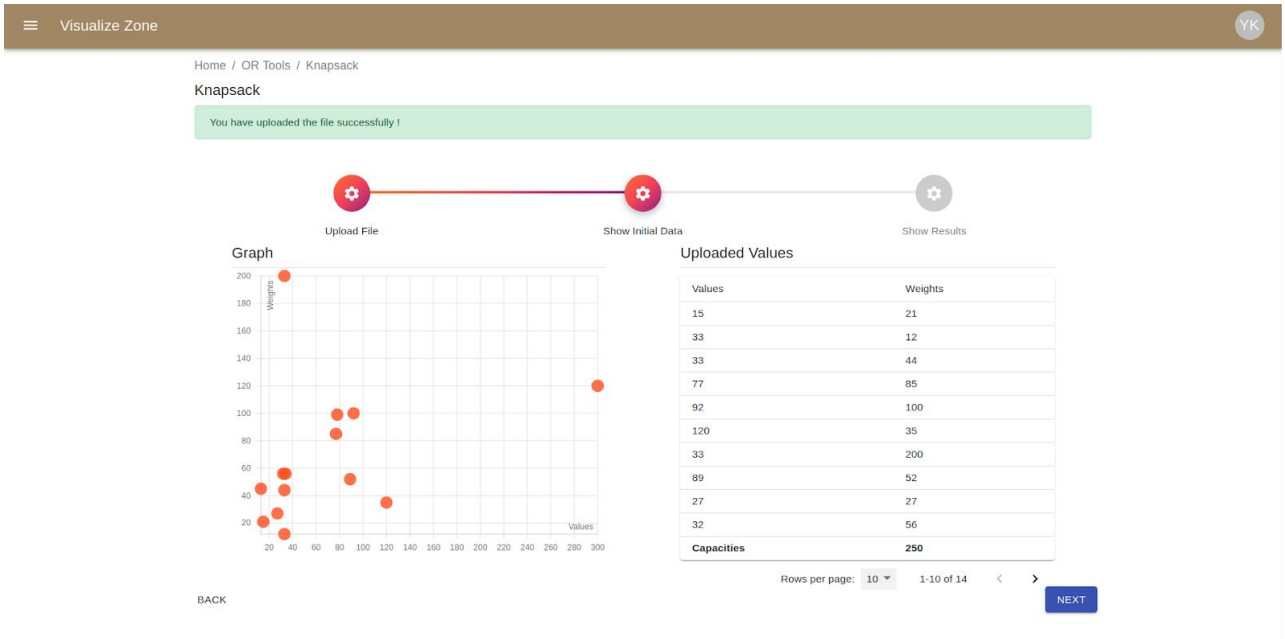
Σε αυτή την εικόνα βλέπουμε την κατάσταση της εφαρμογής μετά από την εισαγωγή ενός αρχείου. Στο αριστερό τμήμα βλέπουμε το πλαίσιο μέσω του οποίου ο χρήστης μπορεί να ανεβάσει ένα αρχείο ή να επιλέξει ένα από τα παραδείγματα που δίνονται από την εφαρμογή για την επίλυση αυτού του αλγορίθμου και την οπτικοποίηση του. Επιπλέον στο δεξί τμήμα βλέπουμε την μορφή που πρέπει να έχει το αρχείο έτσι ώστε να γίνει δεκτό από την εφαρμογή μας. Τέλος στο κάτω τμήμα βλέπουμε τις τιμές του αρχείου που ανεβάσαμε. Τα δεδομένα είναι σελιδοποιημένα και ο χρήστης μπορεί να επιλέξει το πόσα θα βλέπει ανά σελίδα. Πατώντας στο κουμπί “Next” μπορεί να δει τα οπτικοποιημένα δεδομένα.



Εικόνα 35: Ανέβασμα Αρχείου (Knapsack)

6.3.6.3 Οπτικοποίηση Αρχικών δεδομένων (Knapsack)

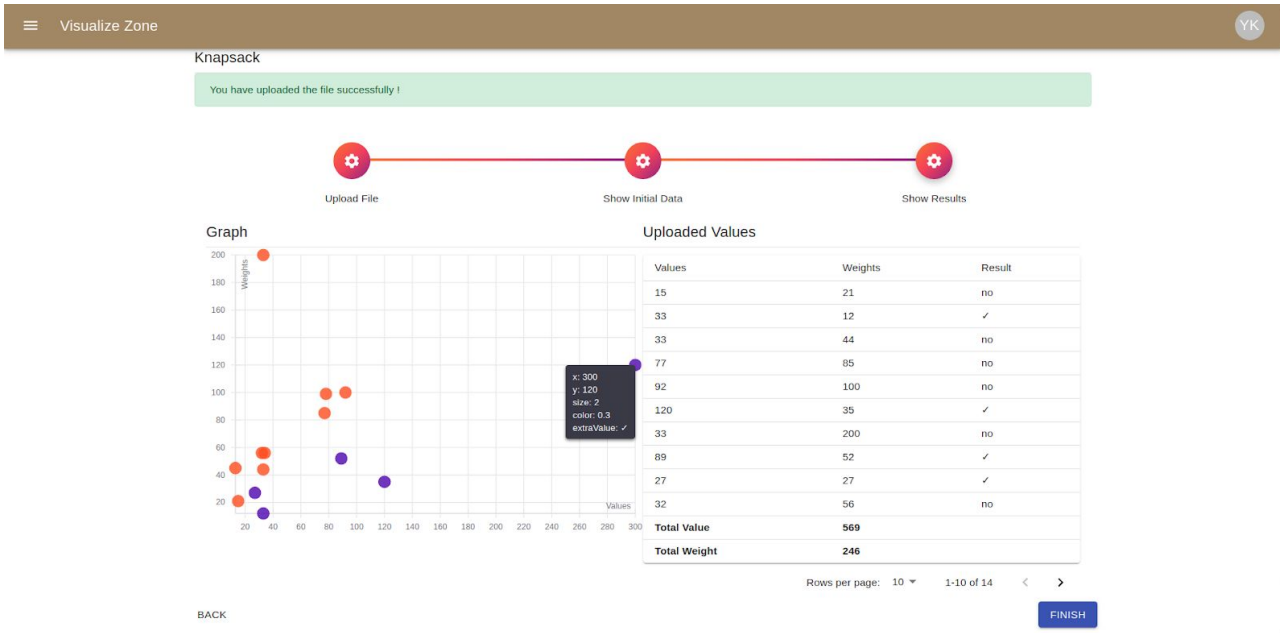
Σε αυτή την εικόνα βλέπουμε την οπτικοποίηση των αρχικών δεδομένων του αλγορίθμου στο αριστερό τμήμα με ένα scatter plot το οποίο έχει ως τετμημένη τις τιμές (values) των αντικειμένων και ως τεταγμένη τα βάρη των αντικειμένων (weights). Στο δεξί τμήμα βλέπουμε τα αρχικά δεδομένα σελιδοποιημένα καθώς και το όριο του βάρους του σάκου που είναι 250 στο συγκεκριμένο παράδειγμα.



Εικόνα 36: Οπτικοποίηση Αρχικών Δεδομένων - Knapsack

6.3.6.4 Οπτικοποίηση Τελικού Αποτελέσματος (Knapsack)

Εδώ βλέπουμε την κατάσταση της εφαρμογής μετά από την εκτέλεση του αλγορίθμου. Μπορούμε να δούμε στον γράφο, σύροντας το ποντίκι μας σε ένα σημείο αν αυτό συμπεριλήφθηκε στο τελικό αποτέλεσμα ή όχι. Ακόμη στον πίνακα βλέπουμε το σύνολο του βάρους αλλά και το άθροισμα των τιμών των αντικειμένων.



Εικόνα 37: Οπτικοποίηση Αποτελεσμάτων - Knapsack

6.3.7 Αλγόριθμος Πολλαπλών Σακιδίων (Multiple Knapsacks)

Ακόμη ένας αλγόριθμος που οπτικοποιήθηκε είναι εκείνος των πολλαπλών σακιδίων. Παρακάτω παρουσιάζεται η διαδικασία οπτικοποίησης αυτών των δεδομένων από το χρήστη.

6.3.7.1 Μορφή αρχείου (Multiple Knapsacks)

Το αρχείο το οποίο θα χρησιμοποιηθεί για την εισαγωγή των αρχικών δεδομένων για την επίλυση του προβλήματος των πολλαπλών σακιδίων έχει την εξής μορφή:

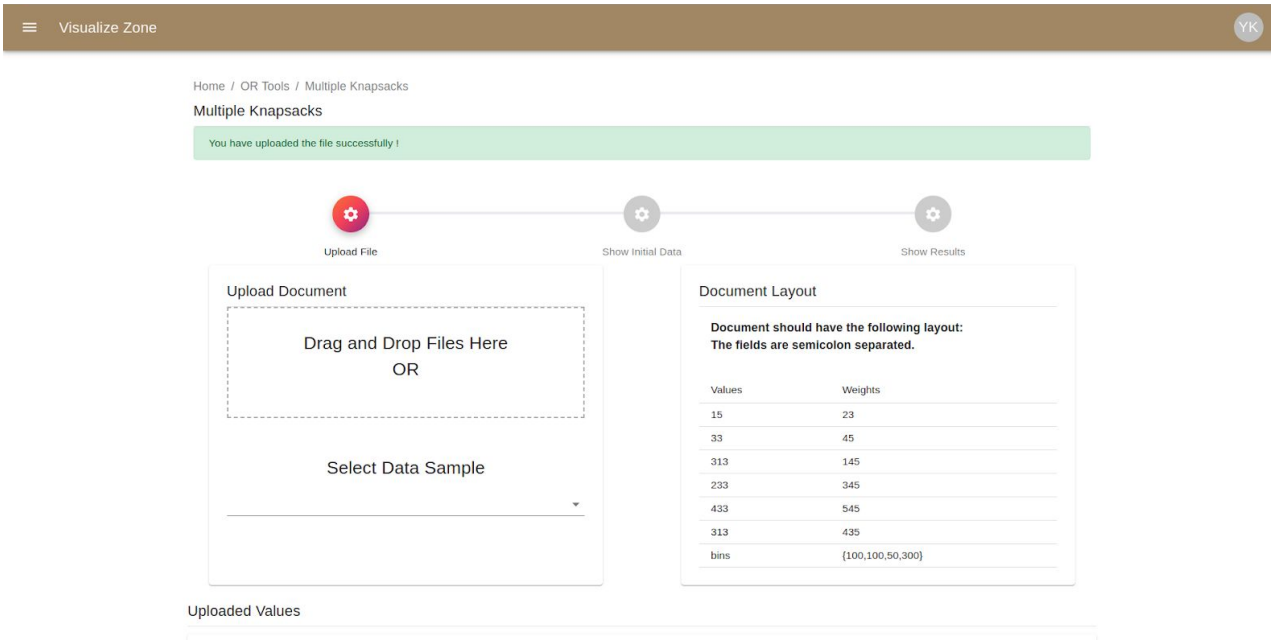
Πίνακας 2: Δεδομένα Αρχείου - Multiple Knapsacks

values	weights
10	48
30	30
25	42
50	36
35	36
30	48
15	42
40	42
30	36
35	24
45	30
10	30
20	42
30	36
25	36
bins	{100,100,100,100,100}

Στην τελευταία γραμμή του συγκεκριμένου αρχείου μέσα στις αγκύλες μπορούμε να βάλουμε το μέγεθος του κάθε σάκου για τον χωρισμό των αντικειμένων. Σε αυτό το παράδειγμα έχουμε 5 σάκους όπου ο καθένας μπορεί να έχει αντικείμενα με άθροισμα βάρους μέχρι 100.

6.3.7.2 Ανέβασμα Αρχείου - Αρχικά Δεδομένα

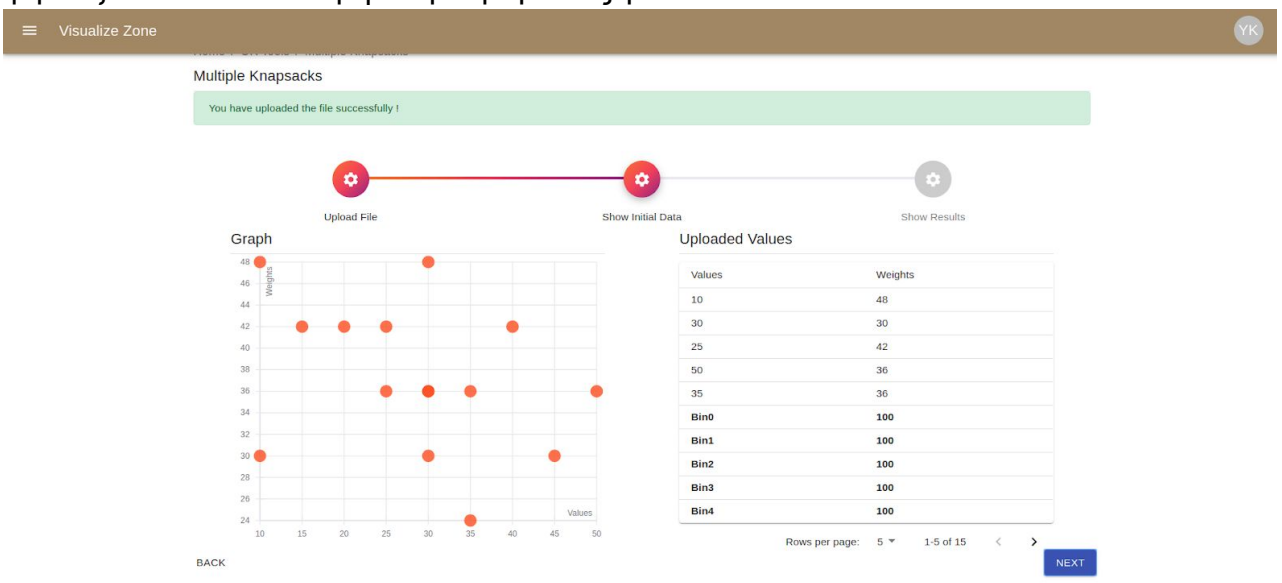
Σε αυτό το τμήμα της εφαρμογής παρουσιάζεται η σελίδα για το ανέβασμα του αρχείου ώστε να δοθούν τα αρχικά δεδομένα για την εκτέλεση του αλγορίθμου των πολλαπλών σακιδίων.



Εικόνα 38: Ανέβασμα Αρχείου - Multiple Knapsacks

6.3.7.3 Οπτικοποίηση Αρχικών δεδομένων

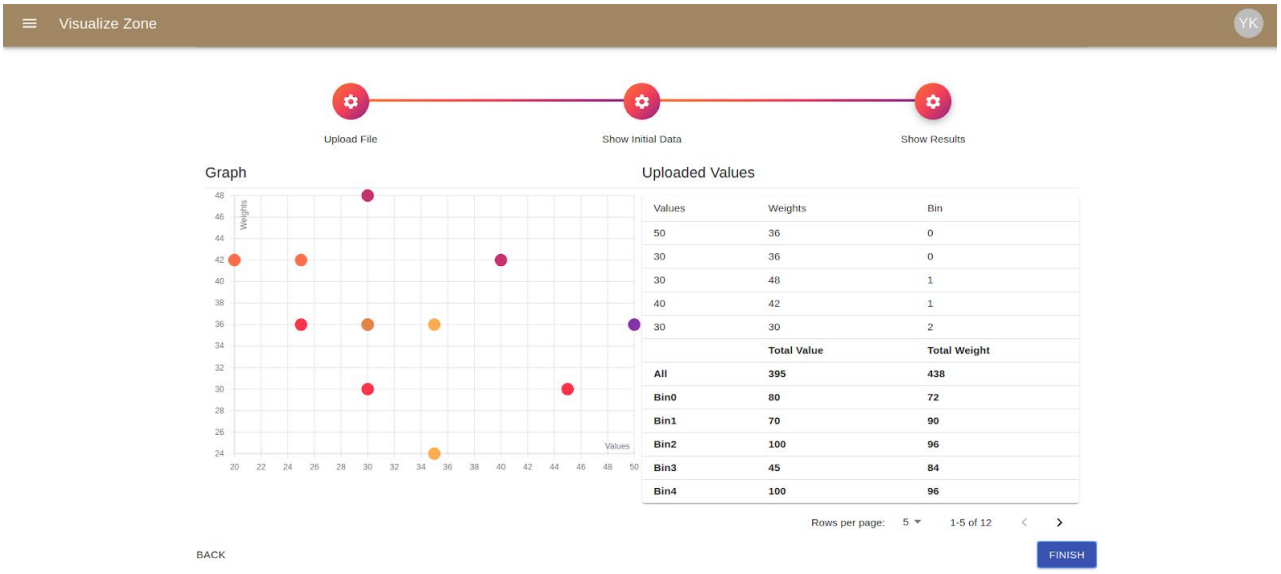
Εδώ βλέπουμε την οπτικοποίηση των αρχικών δεδομένων με ένα scatter plot στο αριστερό τμήμα καθώς και τα αρχικά δεδομένα που έχουν εισαχθεί από το αρχείο. Ακόμη εμφανίζεται και το επιθυμητό όριο μεγέθους για κάθε σάκο.



Εικόνα 39: Οπτικοποίηση αρχικών δεδομένων - Multiple Knapsacks

6.3.7.4 Οπτικοποίηση Τελικού Αποτελέσματος

Το τελικό αποτέλεσμα παρουσιάζεται πάλι με τη μορφή scatter plot. Τα αντικείμενα τα οποία έχουν ομαδοποιηθεί στον κάθε σάκο παρουσιάζονται ως σημεία που έχουν το ίδιο χρώμα ανά ομάδες στο γράφημα. Επιπλέον στο δεξί τμήμα με τον πίνακα μπορούμε να δούμε και το άθροισμα των τιμών αλλά και των βαρών που ομαδοποιήθηκαν σε κάθε σάκο.



Εικόνα 40: Οπτικοποίηση αποτελεσμάτων - Multiple Knapsacks

6.3.8 Αλγόριθμος Περιπλανώμενου Πωλητή (Travelling Salesman Problem)

Ο αλγόριθμος του περιπλανώμενου πωλητή όπως έχει αναφερθεί και παραπάνω έχει υλοποιηθεί με τη βοήθεια του Google Distance API. Γι'αυτό το λόγο ο χρήστης μπορεί να εισάγει μόνο τα ονόματα των πόλεων για να βρεί τη διαδρομή.

6.3.8.1 Μορφή αρχείου (Travelling Salesman)

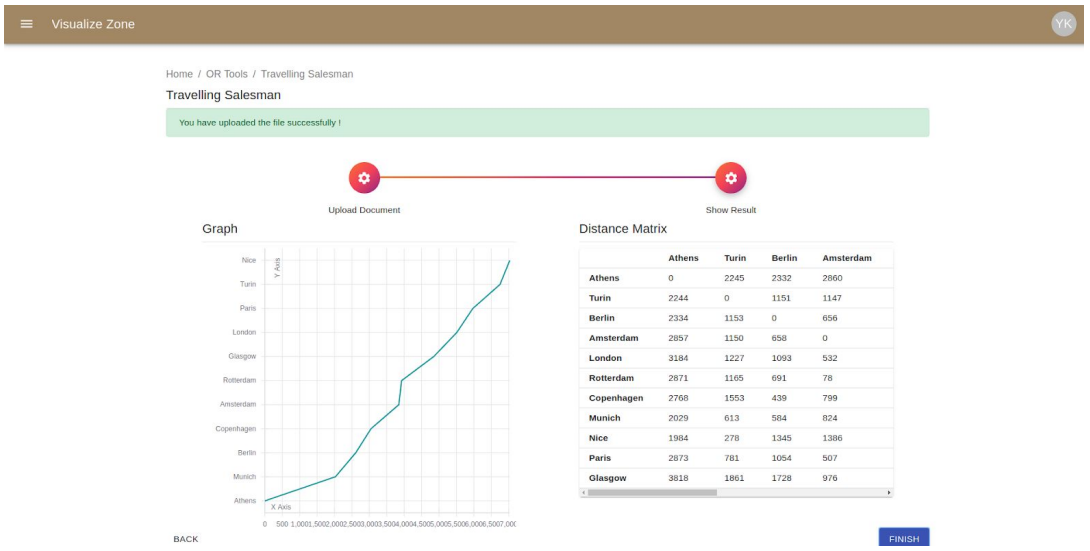
Πίνακας 3: Αρχείο Δεδομένων - Travelling Salesman

cities
Athens
Turin
Berlin
Amsterdam
London
Rotterdam
Copenhagen
Munich
Nice
Paris

6.3.8.2 Οπτικοποίηση Τελικού Αποτελέσματος (Travelling Salesman)

Στο συγκεκριμένο αλγόριθμο έχουμε δύο βήματα για την οπτικοποίηση του αποτελέσματος. Με το δεύτερο βήμα μετά από το ανέβασμα του αρχείου βλέπουμε ένα γράφο γραμμικό ο οποίος παρουσιάζει τις αποστάσεις μεταξύ των πόλεων σε χιλιόμετρα.

Στον άξονα y βλέπουμε τη σειρά με την οποία ο περιπλανώμενος πωλητής θα πρέπει να επισκεφτεί τις πόλεις ενώ στον άξονα x βλέπουμε τις αποστάσεις σε χιλιόμετρα. Στο δεξί τμήμα βλέπουμε τον πίνακα αποστάσεων που έχουμε δημιουργήσει για την επίλυση του.



Εικόνα 41: Οπτικοποίηση Αποτελεσμάτων - Travelling Salesman Problem

6.3.9 Αλγόριθμος Γραμμικής Βελτιστοποίησης (Linear Optimization)

Στα προβλήματα γραμμικής βελτιστοποίησης γίνεται προσπάθεια να βρεθεί μέσα από ένα σύνολο γραμμικών περιορισμών η βέλτιστη τιμή για την αντικειμενική συνάρτηση.

6.3.9.1 Μορφή αρχείου (Linear Optimization)

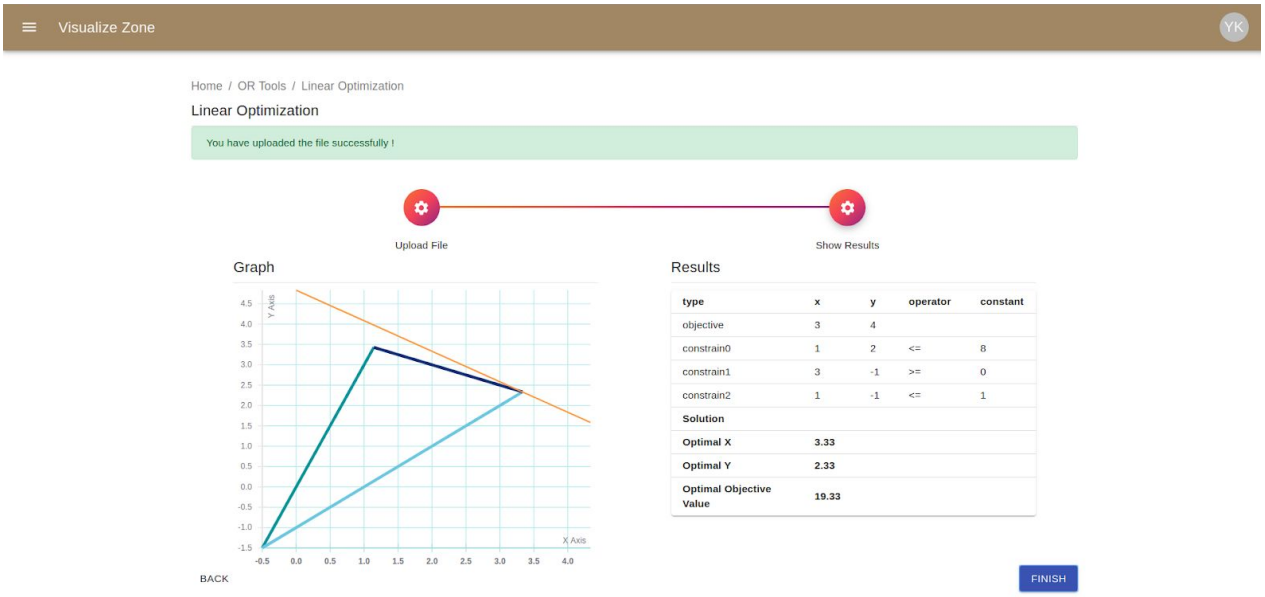
Η μορφή του αρχείου για το πρόβλημα της γραμμικής βελτιστοποίησης παρουσιάζεται παρακάτω. Στην δεύτερη γραμμή παρουσιάζεται η αντικειμενική συνάρτηση που στην περίπτωση μας είναι η 3*x + 4*y. Ενώ στις επόμενες μπορούμε να εισάγουμε τους περιορισμούς που θέλουμε. Για παράδειγμα έχουμε constain0: x*1 + 2 * y <= 8, constain1: 3*x - 1*y >= 0 και τέλος constain2: 1*x - 1*y <= 1.

Πίνακας 4: Αρχείο Δεδομένων - Linear Optimization

type	x	y	operator	constant
objective	3	4		
constrain0	1	2	<=	8
constrain1	3	-1	>=	0
constrain2	1	-1	<=	1

6.3.9.2 Οπτικοποίηση Τελικού Αποτελέσματος (Linear Optimization)

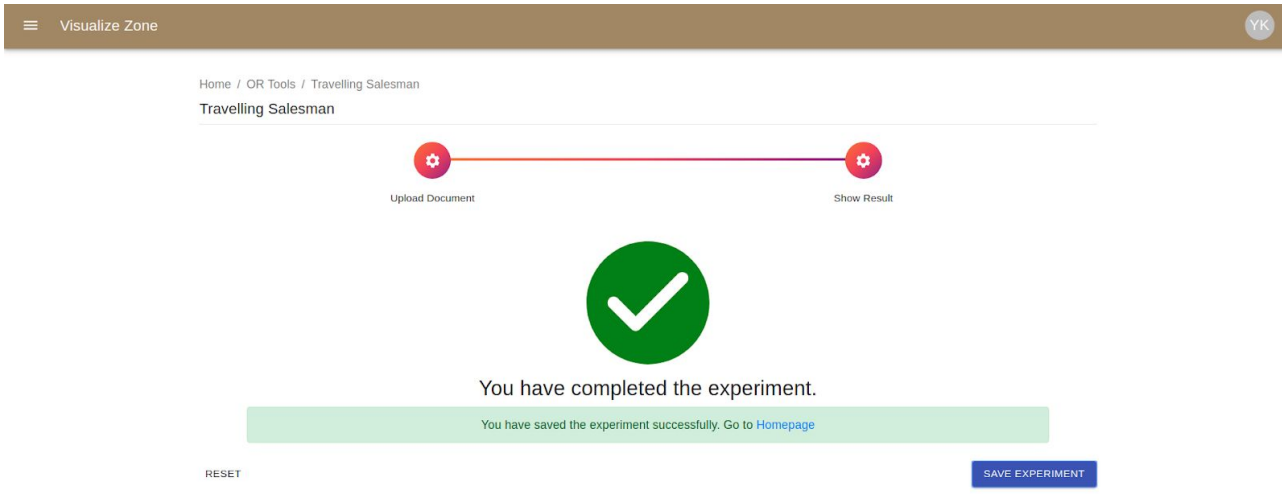
Η οπτικοποίηση του τελικού αποτελέσματος παρουσιάζεται παρακάτω. Στο αριστερό τμήμα βλέπουμε το γράφο που έχει δημιουργηθεί για τους ακόλουθους γραμμικούς περιορισμούς. Επιπλέον στο δεξί τμήμα στον πίνακα παρουσιάζονται οι βέλτιστες τιμές για την τετμημένη και την τεταγμένη της αντικειμενικής συνάρτησης καθώς και η τιμή της.



Εικόνα 42: Οπτικοποίηση Αποτελεσμάτων - Linear Optimization

6.3.10 Αποθήκευση Αποτελέσματος

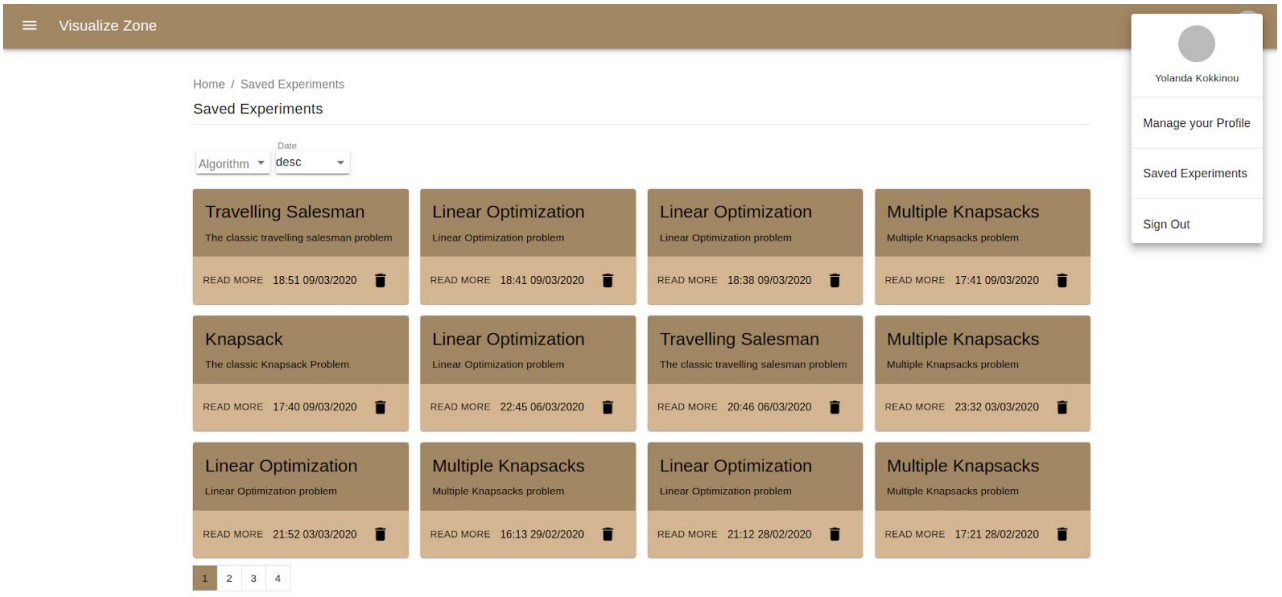
Στο τελευταίο βήμα της εκτέλεσης κάθε αλγορίθμου ο χρήστης έχει την επιλογή να αποθηκεύσει τα δεδομένα του πειράματος που εκτέλεσε για να μπορέσει να τα διαχειριστεί αργότερα.



Εικόνα 43: Αποθήκευση Πειράματος

6.3.11 Αποθηκευμένα Δεδομένα

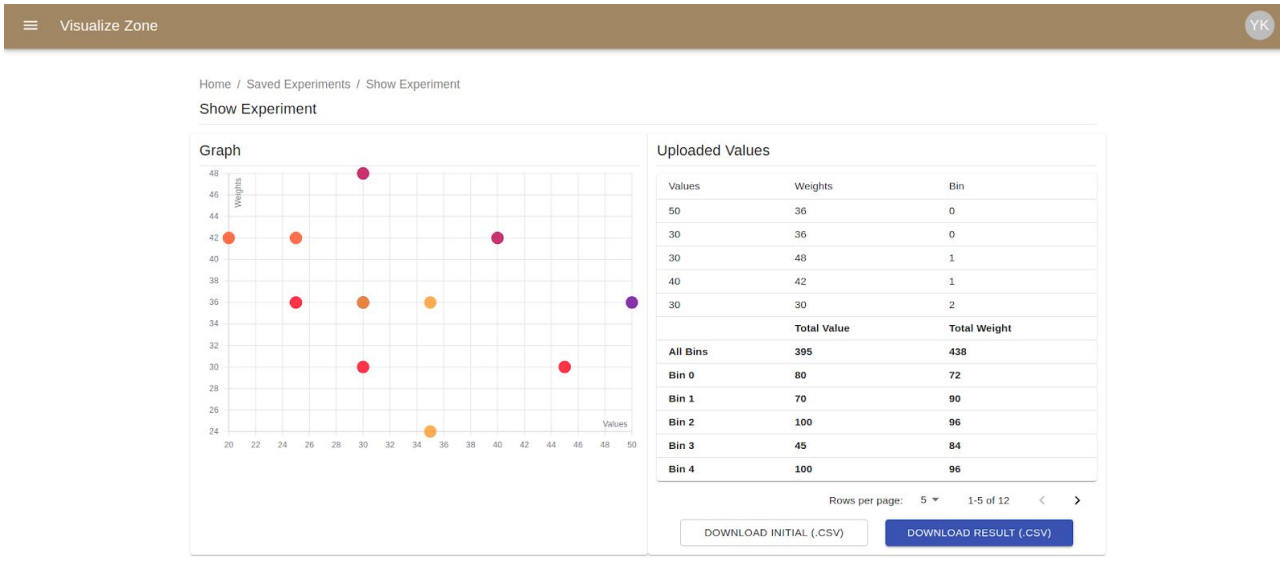
Ο χρήστης μπορεί να διαχειριστεί τα δεδομένα των πειραμάτων που έχει εκτελέσει στο τμήμα της εφαρμογής που λέγεται “My Experiments”. Από εκεί μπορεί να φιλτράρει τα δεδομένα με βάση τον αλγόριθμο ή να τα διατάξει με βάση την ημερομηνία και την ώρα που εκτελέστηκαν. Ακόμη μπορεί να διαγράψει και όσα δεν χρειάζεται πλέον.



Εικόνα 44: Διαχείριση Πειραμάτων

6.3.12 Εμφάνιση Πειραμάτων και κατέβασμα αρχικών και τελικών αρχείων

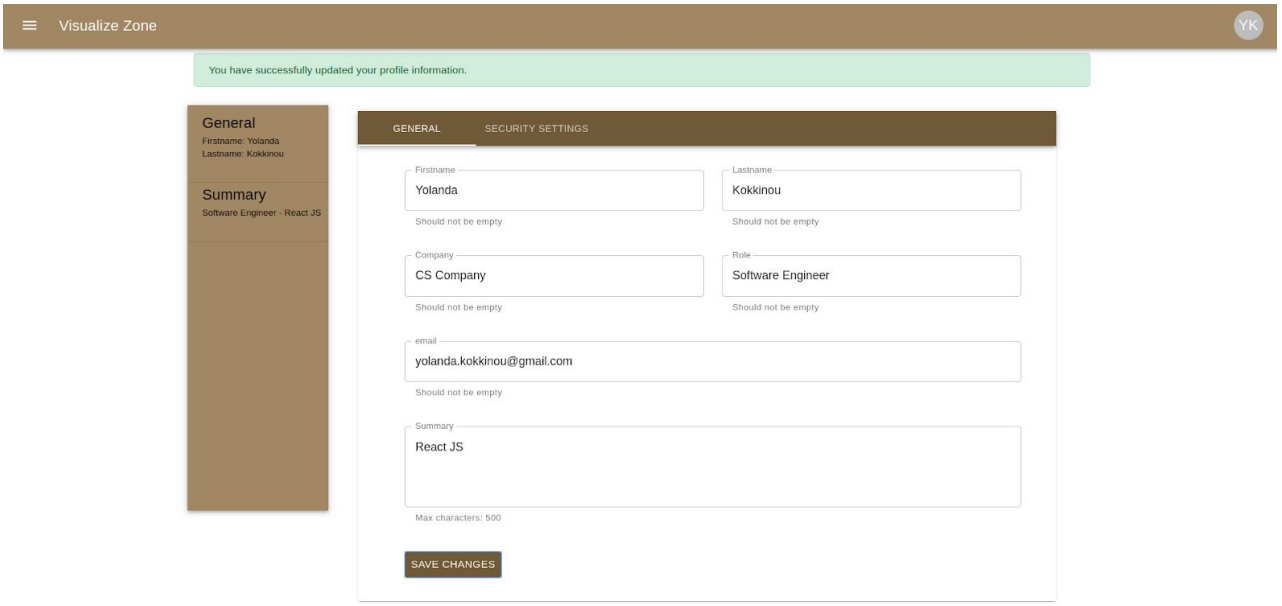
Ο χρήστης κάνοντας κλικ πάνω στις κάρτες που εμφανίζονται στα experiments μπορεί να έχει πρόσβαση στο γράφημα και τα αποτελέσματα του κάθε πειράματος. Επιπλέον μπορεί να κατεβάσει και σε μορφή (.csv) τα αρχεία τα οποία περιέχουν τα αρχικά και τα τελικά δεδομένα. Στο παρακάτω παράδειγμα μπορεί να κατεβάσει τα αρχικά αντικείμενα που θα μπουν στους πολλαπλούς σάκους αλλά και τα αποτελέσματα που εμφανίζονται στον πίνακα δεξιά.



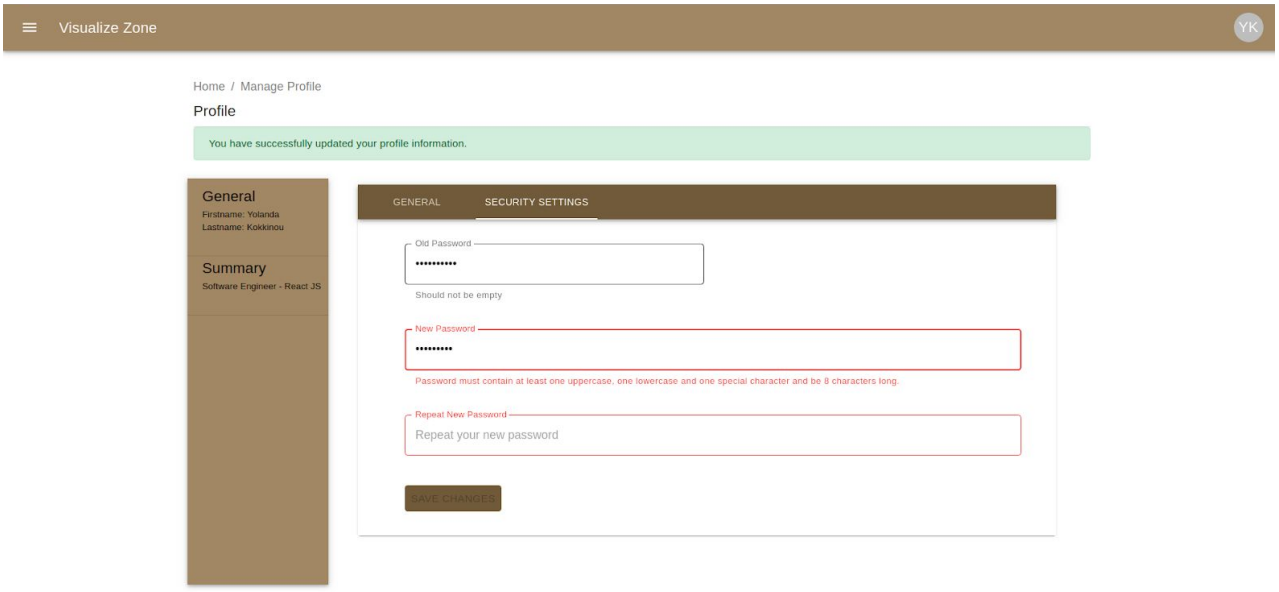
Εικόνα 45: Εμφάνιση Αποτελέσματος - Multiple Knapsacks

6.3.13 Σελίδα Προφίλ

Τέλος ο χρήστης έχει τη δυνατότητα να αλλάξει τα προσωπικά του στοιχεία όπως είναι το όνομα του, που εργάζεται κ.λ.π. Επιπλέον δίνεται και η δυνατότητα να αλλάξει κωδικό πρόσβασης.



Εικόνα 46: Τροποποίηση Προσωπικών Δεδομένων Χρήστη



Εικόνα 47: Αλλαγή Κωδικού

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα πτυχιακή εργασία είχε ως στόχο της προσπάθεια οπτικοποίησης ορισμένων σημαντικών αλγορίθμων της επιστήμης των υπολογιστών και την δημιουργία μιας εφαρμογής για τη διαχείριση τους κυρίως από φοιτητές πληροφορικής αλλά και από διδακτικό προσωπικό.

Για την υλοποίηση της χρησιμοποιήθηκαν αρκετές σύγχρονες τεχνολογίες όπως είναι η βιβλιοθήκη React Vis για την οπτικοποίηση των δεδομένων σε React js. Φυσικά αντιμετωπίστηκαν αρκετά προβλήματα κατά την μοντελοποίηση και την αναπαράσταση των δεδομένων στη συγκεκριμένη μορφή που γίνεται δεκτή από την συγκεκριμένη βιβλιοθήκη του μετωπιαίου άκρου καθώς και από την βιβλιοθήκη Google OR Tools στο νωτιαίο άκρο.

Αποτέλεσε μία ευκαιρία για την εξοικείωση μου με ορισμένους αλγορίθμους αλλά και με το κομμάτι της ανάπτυξης ενός ολοκληρωμένου συστήματος το οποίο να είναι εύκολα συντηρήσιμο και επεκτάσιμο ώστε να φιλοξενήσει και άλλους αλγορίθμους στο μέλλον.

Γενικότερα προσπάθησα να επιτύχω όσο καλύτερη δομή και διαχωρισμό των διαφόρων τμημάτων της εφαρμογής έτσι ώστε να υπάρχει χαλαρή ζεύξη μεταξύ των δύο τμημάτων της εφαρμογής (loose coupling) για να είναι ευκολότερα διαχειρισιμο και επεκτάσιμο το σύστημα.

Κατά την ανάπτυξη του χρησιμοποίησα γνώσεις από διαφορετικά γνωστικά πεδία με τα οποία έχω ασχοληθεί κατά τη διάρκεια των σπουδών μου όπως είναι η δημιουργία εφαρμογών διαδικτύου, η οπτικοποίηση δεδομένων, η επικοινωνία ανθρώπου μηχανής και η ανάπτυξη λογισμικού για αλγοριθμικά προβλήματα. Αποτελεί δηλαδή έναν συνδυασμό αρκετών γνώσεων που πήρα κατά τη διάρκεια των σπουδών μου και δεξιοτήτων.

Τέλος ελπίζω η παρούσα εργασία να αποτελέσει έναυσμα για περαιτέρω έρευνα και πειραματισμό στο συγκεκριμένο κομμάτι της επιστήμης των υπολογιστών έτσι ώστε να έρχονται σε επαφή και οι φοιτητές νωρίτερα με την ανάπτυξη παρόμοιων συστημάτων.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Reliability	Αξιοπιστία
Editor	Επιμελητής
Recommendations	Υποδείξεις
Autoconfiguration	Αυτόματη Παραμετροποίηση
Standalone	Αυτόνομο
Framework	Πλαίσιο Ανάπτυξης
Component	Στοιχείο
Interface	Διεπαφή
Design	Σχεδιασμός
Material	Υλικό
Service	Υπηρεσία
Data	Δεδομένα
Presentation	Παρουσίαση
Dependency Injection	Έγχυση Εξάρτησης
Unified Modelling Language	Ενιαία Γλώσσα Μοντελοποίησης
Knapsack	Σακίδιο
Routing	Δρομολόγηση
Use Case	Περίπτωση Χρήσης
authentication	επαλήθευση
authorization	εξουσιοδότηση
Linear Optimization	Γραμμική βελτιστοποίηση

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
PaaS	Platform as a Service
OR	Operational Research
CI	Continuous Integration
CD	Continuous Deployment
Framework	Πλαίσιο Ανάπτυξης
CRUD	Create, Read, Update, Delete
JAVA EE	JAVA Enterprise Edition
SPA	Single Page Application
API	Application Programming Interface
JPQL	Java Persistence Query Language
DOM	Document Object Model
REST	Representational State Transfer
DAO	Data Access Object
UML	Unified Modelling Language
XML	Extensible Markup Language
JPA	Java Persistence API
ORM	Object Relational Mapping
JAR	Java ARchive
HTTP	HyperText Transfer Protocol
JSON	Javascript Object Notation
JWT	Javascript Web Token

ΑΝΑΦΟΡΕΣ

[1] "Github repository of my undergraduate-thesis", https://github.com/YolandaKok/undergraduate-thesis [Προσπελάστηκε 09/03/2019]
[2] "About OR-Tools" https://developers.google.com/optimization [Προσπελάστηκε 20/12/2019]
[3] "Spring Boot Overview" https://spring.io/projects/spring-boot#overview [Προσπελάστηκε 14/01/2020]
[4] "What is Spring Boot" https://stackify.com/what-is-spring-boot/ [Προσπελάστηκε 14/01/2020]
[5] "Introduction to Apache Maven | A build automation tool for Java projects" https://www.geeksforgeeks.org/introduction-apache-maven-build-automation-tool-java-projects/ [Προσπελάστηκε 14/01/2020]
[6] "Understanding MVC Architecture with React" https://medium.com/createdd-notes/understanding-mvc-architecture-with-react-6cd38e91fefd [Προσπελάστηκε 03/02/2020]
[7] "Component State" https://reactjs.org/docs/faq-state.html [Προσπελάστηκε 04/02/2020]
[8] React in Action, Thomas, M.T. , Manning Publications, 2018
[9] "Spring Data JPA - Reference Documentation" https://docs.spring.io/spring-data/jpa/docs/2.2.0.RELEASE/reference/html/#jpa.query-methods [Προσπελάστηκε 11/02/2020]
[10] Software Architecture Patterns, Mark Richards, O'Reilly, 2015
[11] "React vis - A composable visualization system" https://github.com/uber/react-vis [Προσπελάστηκε 04/03/2020]
[12] "Material Design", https://material.io/design/introduction/#principles [Προσπελάστηκε 04/03/2020]
[13] "Color Tool", https://material.io/resources/color/#!/?view.left=0&view.right=1&primary.color=9F8761 [Προσπελάστηκε 05/03/2020]
[14] "What is Unified Modelling Language(UML)?" https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/ [Προσπελάστηκε 05/03/2020]
[15] "Distance Matrix API - Get Started" https://developers.google.com/maps/documentation/distance-matrix/start [Προσπελάστηκε 06/03/2020]