Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Δικτυακή εφαρμογή για την οπτικοποίηση εξαρτήσεων μεταξύ endpoints μιας διεπαφής REST

Διπλωματικη Εργασια

του

ΔΡΟΣΟΥ ΛΑΛΙΑ

Επιβλέπων: Βασίλειος Βεσκούκης
Καθηγητής ΕΜΠ

Αθήνα, Μάρτιος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Δικτυακή εφαρμογή για την οπτικοποίηση εξαρτήσεων μεταξύ endpoints μιας διεπαφής REST

Διπλωματικη Εργασια

του

ΔΡΟΣΟΥ ΛΑΛΙΑ

Επιβλέπων: Βασίλειος Βεσκούκης
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21 Μαρτίου 2024.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Βασίλειος Βεσκούκης	Νικόλαος Παπασπύρου	Γεώργιος Γκούμας
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

Αθήνα, Μάρτιος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Περίληψη

Στην ψηφιακή εποχή, η αποδοτική διαχείριση και ανάλυση δεδομένων είναι κρίσιμη, ιδιαίτερα στον τομέα ανάπτυξης λογισμικού. Τα Application Programming Interfaces (APIs), λόγω της ευελιξίας και της κλιμακωσιμότητας που προσφέρουν, ικανοποιούν αυτή την ανάγκη, παρέχοντας τα μέσα για την εύκολη επικοινωνία και ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων και πλατφορμών. Για να χρησιμοποιήσουμε όμως ένα REST API σωστά, πρέπει πρώτα να το κατανοήσουμε. Η κατανόηση αυτή επιτρέπει την ομαλή ανάπτυξη και ενσωμάτωση νέων εφαρμογών και υπηρεσιών, ενισχύοντας τη διαλειτουργικότητα και την προσβασιμότητα.

Το API Documentation αποτελεί την τεκμηρίωση που περιγράφει πώς να χρησιμοποιηθεί ένα API, παρέχοντας λεπτομερείς οδηγίες για τα διαθέσιμα endpoints, τα request methods, τις μορφές των δεδομένων που αναμένονται και επιστρέφονται, καθώς και πληροφορίες για τυχόν σφάλματα ή εξαιρέσεις. Είναι ζωτικής σημασίας για τους προγραμματιστές, καθώς διευκολύνει την ορθή χρήση και ενσωμάτωση των APIs σε εφαρμογές και συστήματα. Ωστόσο, πολλές φορές η πληροφορία που περιέχει ένα API Documentation δεν επαρκεί. Συγκεκριμένα, τα διαφορετικά endpoints ενός REST API εμφανίζουν συχνά σχέσεις εξάρτησης μεταξύ τους, η αναγνώριση των οποίων δεν είναι πάντα εύκολη.

Στόχος αυτής της διπλωματικής είναι η βελτίωση της κατανόησης και της χρήσης των APIs μέσω της ανάπτυξης μιας δικτυακής εφαρμογής που εμπλουτίζει την παραγωγή API Documentation οπτικοποιώντας τις εξαρτήσεις μεταξύ των endpoints οι οποίες έχουν αναγνωριστεί με εργαλεία σε προηγούμενες εργασίες [1, 2]. Η οπτικοποίηση των εξαρτήσεων προσφέρει μια καθαρή και διαισθητική κατανόηση του τρόπου αλληλεπίδρασης των διαφορετικών στοιχείων ενός συστήματος. Μέσω της οπτικής αναπαράστασης, οι προγραμματιστές μπορούν πιο εύκολα να αναγνωρίσουν τον τρόπο ροής των δεδομένων μέσα στο σύστημα, να εντοπίσουν πιθανά σημεία συμφόρησης ή ευπάθειας και να βελτιστοποιήσουν την αρχιτεκτονική δημιουργώντας αποδοτικότερα συστήματα.

Λέξεις Κλειδιά

API, REST, SOAP, Postman, API Documentation, Graph visualization, Web application, Javascript, Typescript, React, HTML, NodeJS, MongoDB

Abstract

In the digital era, efficient data management and analysis are critical, especially in the software development sector. Application Programming Interfaces (APIs), due to their flexibility and scalability, meet this need by providing the means for easy communication and data exchange between different systems and platforms.However, in order to use a REST API correctly we first need to understand it. This understanding allows for the smooth development and integration of new applications and services, enhancing interoperability and accessibility.

API Documentation is the documentation that describes how to use an API, providing detailed instructions for the available endpoints, request methods, the formats of expected and returned data, as well as information on any errors or exceptions. It is of vital importance to developers, as it facilitates the correct use and integration of APIs into applications and systems. However, the information contained in API Documentation is not always sufficient. Specifically, the different endpoints of a REST API frequently display dependency relationships among them, which are not always easy to recognize.

The goal of this thesis is to improve the understanding and use of APIs through the development of a web application that enriches the production of API Documentation by visualizing the dependencies among endpoints that have been identified with tools in previous works [1, 2]. Visualizing these dependencies offers a clear and intuitive understanding of how the different elements of a system interact. Through visual representation, developers can more easily recognize the data flow within the system, identify potential congestion points or vulnerabilities, and optimize the architecture to create more efficient systems.

Keywords

API, REST, SOAP, Postman, API Documentation, Graph visualization, Web application, Javascript, Typescript, React, HTML, NodeJS, MongoDB

στους γονείς μου, στην αδερφή μου και στην Ελένη μου

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον αξιότιμο καθηγητή μου κ. Βεσκούκη για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να ασχοληθώ με το παρόν θέμα. Επίσης, ευχαριστώ από καρδιάς τον Υποψήφιο Διδάκτορα και Ερευνητή Χρίστο Χατζιχριστοφή για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς και την αδερφή μου για την καθοδήγηση και υποστήριξη που μου προσέφεραν όλα αυτά τα χρόνια.

Αθήνα, Μάρτιος 2024

Δρόσος Λαλιάς

Περιεχόμενα

Κατάλογος Εικόνων

Κεφάλαιο 1

Εισαγωγή

1.1 Θέμα

Η τεκμηρίωση των APIs είναι ζωτικής σημασίας για την αποτελεσματική ανάπτυξη και ενσωμάτωση λογισμικού, καθώς παρέχει τις απαραίτητες οδηγίες για τη χρήση ενός API. Το θέμα της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας εφαρμογής οπτικοποίησης του γραφήματος εξαρτήσεων για APIs, με σκοπό τη βελτίωση της κατανόησης των σχέσεων και των εξαρτήσεων μεταξύ των endpoints που έχουν εντοπιστεί με εργαλεία από προηγούμενες διπλωματικές στο εργαστήριο. Η δημιουργία της εφαρμογής αυτής κινήθηκε από την ανάγκη να καταστούν οι περίπλοκες διασυνδέσεις μεταξύ διαφορετικών στοιχείων ενός API πιο κατανοητές, διευκολύνοντας έτσι την ανάπτυξη και τη συντήρηση λογισμικού.

1.2 Δομή Πτυχιακής Εργασίας

Η εργασία αυτή είναι οργανωμένη σε τέσσερα κεφάλαια:

1. Στο κεφάλαιο 2 πραγματοποιείται λεπτομερής ανάλυση των REST APIs και των υπάρχοντων εργαλείων ανάλυσης και εξαγωγής εξαρτήσεων APIs.
2. Στο κεφάλαιο 3 περιγράφεται αναλυτικά η λειτουργία και η υλοποίηση της εφαρμογής.
3. Στο κεφάλαιο 4 διενεργείται μία σύντομη αναφορά στα συμπεράσματα και τις μελλοντικές επεκτάσεις αυτής της εργασίας.

Κεφάλαιο 2

APIs: Ορισμός και Εφαρμογές

Στη σημερινή εποχή της ψηφιακής υπερπληροφόρησης, όπου τα δεδομένα αναπαράγονται με καταιγιστικούς ρυθμούς, η αξία της διαχείρησης της πληροφορίας συνεχώς ενισχύεται. Η ανάπτυξη και χρήση τεχνολογιών που διευκολύνουν την διαχείριση δεδομένων και την επικοινωνία μεταξύ εφαρμογών, με στόχο τον περιορισμό της προγραμματιστικής πολυπλοκότητας, επιβάλλεται. Παρόλο που πληθώρα τεχνολογιών έχει αναπτυχθεί για αυτόν τον σκοπό, τα APIs ξεχωρίζουν λόγω της απλότητας και της ευελιξίας τους, καθιστώντας τα αναντικατάστατο εργαλείο στον ψηφιακό κόσμο της ανάπτυξης λογισμικού.

2.1 Ιστορία του API - Ορισμός

Η ιστορία των API (Application Programming Interfaces - Διεπαφές Προγραμματισμού Εφαρμογών) αντιπροσωπεύει μια σημαντική εξέλιξη στον τομέα της πληροφορικής και της τεχνολογίας [3]. Τα API επιτρέπουν την αλληλεπίδραση μεταξύ διαφορετικών συστημάτων και εφαρμογών, επιτρέποντας την ανταλλαγή δεδομένων και λειτουργιών με τρόπο αυτοματοποιημένο και αποδοτικό.

Η χρήση των API δεν είναι νέα· υπήρχαν από τις αρχές της δεκαετίας του 1960, όταν οι προγραμματιστές αναζητούσαν τρόπους για την απλούστευση της αλληλεπίδρασης μεταξύ των προγραμμάτων λογισμικού. Ωστόσο, η πραγματική τους ανάδειξη ξεκίνησε στα τέλη της δεκαετίας του 1990 και την αρχή του 2000, με την έλευση του Διαδικτύου και των web εφαρμογών. Η δημιουργία των πρώτων web APIs, όπως το SOAP (Simple Object Access Protocol) και αργότερα το REST (Representational State Transfer), άλλαξε δραματικά τον τρόπο ανάπτυξης και αλληλεπίδρασης των εφαρμογών στο Διαδίκτυο.

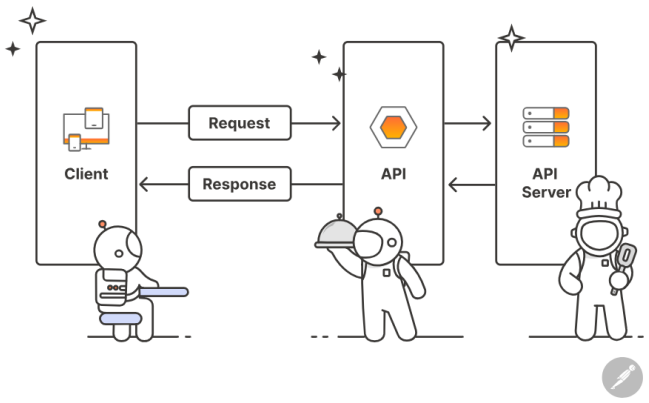
Η εμφάνιση του REST στην αρχή της δεκαετίας του 2000 σημάδεψε μια σημαντική στιγμή, καθώς προσέφερε μια απλούστερη και πιο ευέλικτη εναλλακτική στο SOAP, επιτρέποντας ευκολότερη ανάπτυξη και ταχύτερη επικοινωνία μεταξύ των εφαρμογών στον ιστό. Το REST χρησιμοποιεί το HTTP πρωτόκολλο για την ανταλλαγή δεδομένων, κάτι που το καθιστά ιδιαίτερα προσβάσιμο και ευρέως υιοθετηθέν.

Ακολούθησε η εποχή των μικροϋπηρεσιών και των cloud υπηρεσιών, κατά τη διάρκεια της οποίας τα API έγιναν ακόμη πιο κρίσιμα. Επέτρεψαν στις επιχειρήσεις να υιοθετήσουν πιο ευέλικτες και κλιμακούμενες αρχιτεκτονικές, βελτιώνοντας την ικανότητά τους να ανταποκρίνονται γρήγορα σε αλλαγές της αγοράς και να παρέχουν νέες υπηρεσίες.

Σήμερα, τα API είναι αναπόσπαστο μέρος του οικοσυστήματος τεχνολογίας, επιτρέποντας

την άμεση ενσωμάτωση και συνεργασία μεταξύ εφαρμογών, υπηρεσιών και συστημάτων, από την ανάπτυξη εφαρμογών κινητής τηλεφωνίας μέχρι τη διαχείριση εταιρικών εφαρμογών.

Το API [4], είναι η διεπαφή μέσω της οποίας τα δεδομένα τα οποία καταχωρούμε, σε μία εφαρμογή ή πλατφόρμα, αποστέλλονται σε έναν server και στέλνονται ξανά πίσω σε εμάς, με τις απαντήσεις και τα αποτελέσματα που επιθυμούμε. Ένα παράδειγμα για καλύτερη κατανόηση του πώς ακριβώς λειτουργεί το API, είναι οι πλατφόρμες που λειτουργούν ως μεσάζοντες των e-shops κάθε επιχείρησης (πχ Skroutz). Ο χρήστης μπαίνει στην πλατφόρμα και καταχωρεί τα δεδομένα της αναζήτησής που επιθυμεί. Στη συνέχεια, η πλατφόρμα επικοινωνεί με την ιστοσελίδα της κάθε επιχείρησης ώστε να δώσει πίσω τις πληροφορίες που χρειάζεται κάποιος. Επίσης, ένα άλλο γνωστό παράδειγμα του πώς χρησιμοποιείται το API, αποτελεί και το Amazon Web Services (AWS).



Εικόνα 2.1: *Τρόπος λειτουργίας ενός API*

Υπάρχουν 4 βασικοί τύποι API [4]:

1. Open API: Το Open API είναι δημόσιο και ανοιχτό προς χρήση για όλους με το πρωτόκολλο HTTP και δεν υπάρχει κάποιος περιορισμός στην πρόσβαση σε αυτό.

2. Partner API: Το Partner API ή αλλιώς API συνεργατών περιλαμβάνει συγκεκριμένες άδειες που πρέπει να έχει κάποιος για να έχει πρόσβαση σε αυτό.

3. Internal API: Το Internal API ή εσωτερικό API δημιουργείται εσωτερικά από μία εταιρεία για εταιρική χρήση.

4. Composite API: Ο τέταρτος τύπος είναι το Composite API, το οποίο συνδυάζει διαφορετικά API δεδομένων και υπηρεσιών.

2.2 REST APIs

2.2.1 Ορισμός και αρχές

Τα REST APIs [5](Representational State Transfer Application Programming Interfaces) είναι διεπαφές προγραμματισμού εφαρμογών που ακολουθούν την αρχιτεκτονική REST. Τα REST APIs επιτρέπουν την αλληλεπίδραση μεταξύ των εφαρμογών και των web services με έναν τρόπο που είναι εύκολο να κατανοηθεί και να χρησιμοποιηθεί.

Η λειτουργία των REST APIs στηρίζεται στις ακόλουθες βασικές αρχές:

1. Resources: Στο επίκεντρο της αρχιτεκτονικής REST βρίσκονται οι πόροι, οι οποίοι μπορούν να είναι δεδομένα, αντικείμενα ή υπηρεσίες που είναι διαθέσιμα για αλληλεπίδραση μέσω του διαδικτύου.

2. Uniform Interface: Η ενιαία διεπαφή παρέχει έναν σταθερό και προβλέψιμο τρόπο αλληλεπίδρασης με τους πόρους. Αυτό περιλαμβάνει τη χρήση τυπικών μεθόδων HTTP (όπως GET, POST, PUT, DELETE) για την ανάκτηση, δημιουργία, τροποποίηση ή διαγραφή πόρων.

3. Client-Server Decoupling: Τα REST APIs επιτρέπουν την αποσύνδεση της πελατειακής λογικής από τη λογική του διακομιστή, προάγοντας έτσι την ευελιξία και την επεκτασιμότητα των εφαρμογών.

4. Stateless Communication: Κάθε αίτημα από τον πελάτη προς τον διακομιστή πρέπει να περιέχει όλες τις πληροφορίες που ο διακομιστής χρειάζεται για να καταλάβει και να εκτελέσει το αίτημα. Ο διακομιστής δεν πρέπει να αποθηκεύει κατάσταση πελάτη μεταξύ αιτημάτων.

5. Cacheability: Η δυνατότητα αποθήκευσης αποκρίσεων σε cache μειώνει την ανάγκη για επαναλαμβανόμενα αιτήματα, βελτιώνοντας την απόδοση και την κλιμάκωση των εφαρμογών.

6. Interoperability: Τα REST APIs είναι platform-independent και μπορούν να υλοποιηθούν σε οποιαδήποτε γλώσσα προγραμματισμού. Οι clients μπορούν εύκολα να τα χρησιμοποιήσουν σε διαφορετικές τεχνολογίες, πράγμα που οδηγεί σε αυξημένη διαλειτουργικότητα.

2.2.2 HTTP Methods

Για να επικοινωνήσει κάποιος με ένα API, πρέπει να κάνει κάποιο HTTP request προς αυτό. Τα πέντε βασικά HTTP methods που χρησιμοποιούνται για την επικοινωνία και στην ανάπτυξη των RESTful APIs είναι [6]:

1. POST: Χρησιμοποιείται για τη δημιουργία ενός πόρου σε μια συλλογή.

```
HTTP

POST /test HTTP/1.1
Host: foo.example
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

field1=value1&field2=value2
```

Εικόνα 2.2: *Παράδειγμα POST request*

2. PUT: Ανάλογο του POST method, χρησιμοποιείται για την ενημέρωση ολόκληρου του περιεχομένουν ενός ή περισσότερων πόρων μιας συλλογής.

```
PUT /new.html HTTP/1.1
Host: example.com
Content-type: text/html
Content-length: 16

<p>New File</p>
```

Εικόνα 2.3: *Παράδειγμα PUT request*

3. PATCH: Χρησιμοποιείται για την ενημέρωση πόρων, αλλά αντίθετα με το PUT, τροποποιεί μόνο μέρος των περιεχομένων ενός πόρου και όχι ολόκληρο το περιεχόμενο.

```
PATCH /file.txt HTTP/1.1
Host: www.example.com
Content-Type: application/example
If-Match: "e0023aa4e"
Content-Length: 100

[description of changes]
```

Εικόνα 2.4: *Παράδειγμα GET request*

4. GET: Είναι η πιο συνηθισμένη μέθοδος και χρησιμοποιείται για την επιστροφή των δεδομένων ενός πόρου.

Εικόνα 2.5: *Παράδειγμα GET request*

5. DELETE: Αυτή η μέθοδος χρησιμοποιείται για την αφαίρεση ενός πόρου.

```
DELETE /file.html HTTP/1.1
Host: example.com
```

Εικόνα 2.6: *Παράδειγμα DELETE request*

Αυτές οι μεθόδοι είναι θεμελιώδεις για την αλληλεπίδραση με πόρους και συλλογές πόρων σε ένα REST περιβάλλον, παρέχοντας τα μέσα για τη δημιουργία, την ενημέρωση, την ανάγνωση και τη διαγραφή πόρων.

2.2.3 Parameter Types

Κάθε request που γίνεται με κάποια από τις παραπάνω μεθόδους περιλαμβάνει και κάποιες παραμέτρους οι οποίες ορίζουν με ποιους πόρους θέλουμε να αλληλεπιδράσουμε. Οι πιο κοινοί τύποι παραμέτρων που χρησιμοποιούνται σε ένα REST API, είναι [7]:

1. Path Parameters: Τα path parameters εμφανίζονται στο τέλος του URL, πριν από τον χαρακτήρα «;» και περιλαμβάνουν μόνο την τιμή του attribute. Χρησιμοποιούνται κυρίως σε GET, PATCH και PUT μεθόδους.

```
/service/myresource/user/{user}/bicycles/{bicycleId}
```

Εικόνα 2.7: *Παράδειγμα Path parameters*

2. Query String Parameters: Τα query string parameters εμφανίζονται στο τέλος του URL, χωρίζονται από αυτό με τον χαρακτήρα «;» και περιλαμβάνουν το attribute μαζί με την τιμή που δίνουμε ως παράμετρο. Χρησιμοποιούνται κυρίως σε GET μεθόδους με σκοπό το filtering ή το sorting των αποτελεσμάτων του request

```
?myparam1=123&myparam2=abc&myparam2=xyz
```

Εικόνα 2.8: *Παράδειγμα Query string parameters*

3. Header Parameters: Τα header parameters χρησιμοποιούνται σε όλα τα HTTP methods για να παρέχουν πρόσθετες πληροφορίες στην αίτηση ή στην απόκριση του API. Συνήθως περιέχουν πιστοποιητικά ασφαλείας, tokens ή άλλα δεδομένα που βοηθούν στην επαλήθευση της ταυτότητας του χρήστη ή της εφαρμογής που κάνει την κλήση.

Accept-Ranges:	bytes
Access-Control-Allow-Origin:	*
Age:	38840
Alt-Svc:	h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
Cache-Control:	public, max-age=31536000
Content-Length:	15920

Εικόνα 2.9: *Παράδειγμα Header parameters*

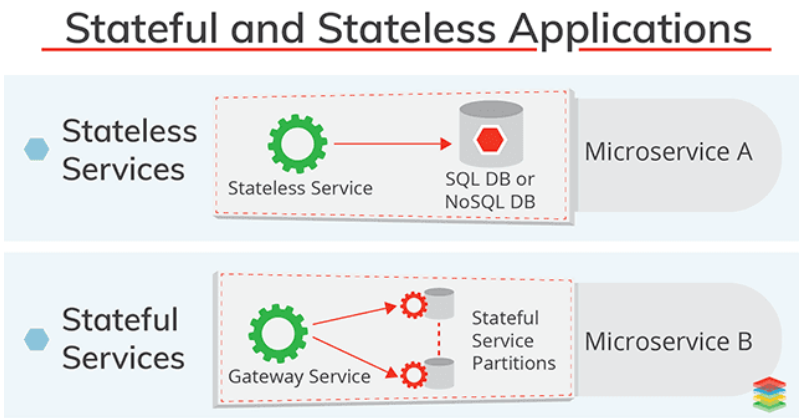
4. Request Body Parameters: Με κάθε request μεθόδου POST PUT ή PATCH συνήθως απαιτείται ένα request body. Τα request bodies αποτελούν ολοκληρωμένα JSON objects και περιέχουν την απαραίτητη πληροφορία για την επιτυχή επικοινωνία με το API.

```
myparam1=123&myparam2=abc&myparam2=xyz
```

Εικόνα 2.10: *Παράδειγμα Body parameters*

2.3 Stateful vs Stateless API

Υπάρχουν δύο κατηγορίες στις οποίες χωρίζονται τα APIs: stateful και stateless [1]. Σε ένα stateful API ο server διατηρεί όλη την πληροφορία για την κατάσταση της συνεδρίας του client, δηλαδή περιέχει την απαραίτητη πληροφορία για την ταυτοποίηση του client και την κατάσταση που βρίσκεται κάθε στιγμή. Με αυτό τον τρόπο επιτρέπει στον χρήστη να χρησιμοποιεί την υπηρεσία αδιάκοπα καθώς δεν χρειάζεται να ξαναστέλνει την ίδια πληροφορία σε κάθε request που εκτελεί.



Εικόνα 2.11: *Stateful vs Stateless*

Σε ένα stateless API, κάθε request που εκτελεί ένας χρήστης περιέχει όλη την πληροφορία για την ταυτοποίηση του και την παρούσα κατάσταση της συνεδρίας. Ο server επεξεργάζεται κάθε request ξεχωριστά και δεν αποθηκεύει καμία πληροφορία σχετική με την συνεδρία του client.

Κάθε κατηγορία εμφανίζει πλεονεκτήματα και μειονεκτήματα. Ένα stateful API εμφανίζει καλύτερη επίδοση καθώς ο server αποθηκεύει όλη την πληροφορία η οποία μπορεί

να αξιοποιηθεί από τους προγραμματιστές για να την προσαρμόσουν στις ανάγκες της εφαρμογής τους και να δημιουργήσουν αξιόπιστες εφαρμογές χωρίς να στηρίζονται σε τρίτους. Ακόμη επιτρέπει ενημερώσεις σε ζωντανό χρόνο λόγω της γρήγορης ανταπόκρισης των εφαρμογών, ενώ προσφέρει και μια αδιάκοπη εμπειρία χρήσης στους καταναλωτές. Ωστόσο, πέρα από το ότι είναι πιο περίπλοκα από ένα stateless API, είναι επίσης λιγότερο κλιμακώσιμα λόγω της πληροφορίας που αποθηκεύουν σε κάθε request και μπορεί να δημιουργήσουν καθυστερήσεις στην ανταπόκριση σε περίπτωση που υπάρχουν πολλαπλά requests. Τέλος, χρησιμοποιούν μεγαλύτερο εύρος ζώνης στο δίκτυο καθώς απαιτείται μεγαλύτερη ανταλλαγή πληροφοριών μεταξύ του client και του server. Ένα χαρακτηριστικό παράδειγμα stateful API είναι το SOAP API, το οποίο όμως μπορεί να γίνει και stateless.

Από την άλλη, τα stateless APIs είναι πιο εύκολα και γρήγορα στην υλοποίηση από τα stateful APIs. Είναι cacheable, δηλαδή μπορούν να αποθηκευτούν σε τοπική cache για ταχύτερη πρόσβαση σε αυτά ενώ εμφανίζουν και μεγαλύτερη συμβατότητα με άλλες εφαρμογές χωρίς να απαιτούν ιδιαίτερη προσπάθεια διατήρησης. Στα αρνητικά τους είναι ότι εμφανίζουν γενικά χαμηλότερη επίδοση σε σχέση με τα stateful APIs και δεν προτείνονται για εφαρμογές που απαιτούν συχνές ενημερώσεις σε πραγματικό χρόνο.

Στα stateful APIs ο server ανοίγει τη συνεδρία και ο server την τερματίζει. Όλη η πληροφορία είναι κλειδωμένη στη συνεδρία και ο server έχει την ευθύνη να την διατηρεί. Αντίθετα, στα stateless APIs η ευθύνη μεταφέρεται στον client και ο φόρτος διαμοιράζεται. Για παράδειγμα, σε ένα stateless API που πραγματοποιεί παραγγελίες βιβλίων, ο server δεν χρειάζεται να συγκρατεί το id του καροτσιού του πελάτη αλλά ο client είναι αυτός που στέλνει σε κάθε request το id, γεγονός που ελαφρύνει σημαντικά τον φόρτο του server. Ένα χαρακτηριστικό παράδειγμα stateless API είναι το REST API το οποίο περιγράψαμε προηγουμένως.

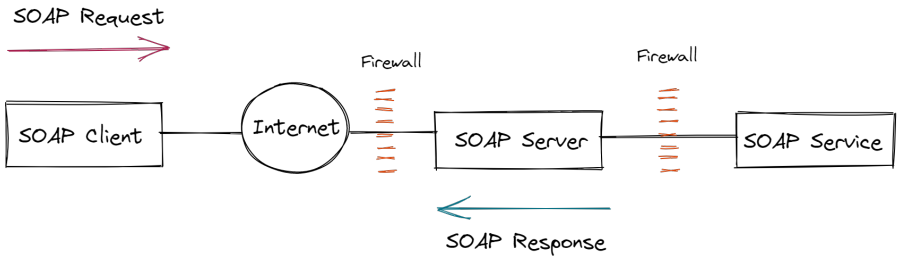
2.3.1 SOAP

Το SOAP (Simple Object Access Protocol) [8] API είναι ένα πρωτόκολλο επικοινωνίας που επιτρέπει στα προγράμματα να ανταλλάσσουν πληροφορίες μέσω του διαδικτύου. Το SOAP χρησιμοποιεί XML για τη διαμόρφωση των μηνυμάτων επικοινωνίας, εξασφαλίζοντας έτσι μια αυστηρή δομή στην ανταλλαγή δεδομένων. Αυτό το καθιστά ιδιαίτερα κατάλληλο για επίσημες επιχειρησιακές εφαρμογές και περιπτώσεις χρήσης που απαιτούν εξασφαλισμένες συναλλαγές και υψηλά επίπεδα ασφάλειας.

Το SOAP είναι ανεξάρτητο από την πλατφόρμα και το πρωτόκολλο μεταφοράς, πράγμα που σημαίνει ότι μπορεί να λειτουργήσει πάνω από διάφορα δίκτυα και πρωτόκολλα όπως HTTP, SMTP, TCP κ.α., καθιστώντας το ευέλικτο για ποικίλες εφαρμογές. Επιπρόσθετα, το SOAP υποστηρίζει την πλήρη δυνατότητα WS-* προτύπων, που περιλαμβάνουν ασφάλεια και αξιοπιστία, παρέχοντας πρόσθετες δυνατότητες για την ανάπτυξη περίπλοκων επιχειρησιακών εφαρμογών.

Η χρήση του SOAP API απαιτεί την κατανόηση των WSDL (Web Services Description Language) αρχείων, τα οποία περιγράφουν τις διεπαφές των διαθέσιμων web υπηρεσιών. Αυτά τα αρχεία WSDL διευκολύνουν την αυτόματη δημιουργία client και την εύκολη ενσωμάτωση μεταξύ διαφορετικών συστημάτων.

Παρά τις προκλήσεις που παρουσιάζει όσον αφορά την πολυπλοκότητα και την απαιτούμενη περισσότερη προσπάθεια για την ανάπτυξη σε σύγκριση με άλλες προσεγγίσεις όπως το REST, το SOAP παραμένει μια σημαντική επιλογή για την ασφαλή και αξιόπιστη ανταλλαγή δεδομένων σε επιχειρησιακό επίπεδο, ειδικά σε τομείς όπως οι χρηματοοικονομικές υπηρεσίες, η υγειονομική περίθαλψη και η κυβερνητική διοίκηση.



Εικόνα 2.12: *SOAP Request Example*

2.3.2 SOAP vs REST

Όταν αντιμετωπίζουμε καταστάσεις που απαιτούν επανειλημμένες επικοινωνίες με μια υπηρεσία για την εκτέλεση μιας διαδικασίας, το SOAP αποτελεί την καλύτερη επιλογή, καθώς διαχειρίζεται αποτελεσματικά πολλαπλές αιτήσεις υπό τον έλεγχο του διακομιστή. Με την υποστήριξη των WS-security και WS-AtomicTransactions, προσφέρει μεγαλύτερη ασφάλεια σε σύγκριση με το REST, κάτι που το καθιστά ιδανικό για εφαρμογές με αυξημένες απαιτήσεις αξιοπιστίας, όπως είναι οι τραπεζικές και οικονομικές εφαρμογές. Επιπλέον, σε περίπτωση αποτυχίας μιας συναλλαγής, το SOAP λαμβάνει την πρωτοβουλία για αυτόματη επανάληψη, εν αντιθέσει με το REST που απαιτεί παρέμβαση από τον πελάτη. Αντίθετα, εάν οι συναλλαγές δεν αποτελούν μέρος των απαιτήσεων και η επικέντρωση δεν είναι στην αξιοπιστία, το REST API και γενικά τα stateless APIs μπορούν να είναι προτιμότερα χάρη στην απλότητα και κλιμακωσιμότητά τους, προσφέροντας μια άριστη λύση για εταιρείες που επιδιώκουν την ταχεία ανάπτυξη και ευελιξία των εφαρμογών τους. Η ευκολία στην ανάπτυξη και η συμβατότητα με διάφορα συστήματα και εφαρμογές ενισχύουν τη δημοτικότητα του REST API, παρά το γεγονός ότι το SOAP παραμένει η επικρατέστερη επιλογή για επιχειρηματικές εφαρμογές λόγω της ασφάλειας που προσφέρει. [9, 10]

2.4 Εργαλεία ανάλυσης APIs και εξαγωγής εξαρτήσεων

2.4.1 Εργαλεία για την παραγωγή API Documentation

Πολλές προσπάθειες έχουν γίνει για την ανάλυση και την αυτόματη παραγωγή τεκμηρίωσης ενός API, καθώς είναι αυτό που μπορεί να αναδείξει την αξία του. Στον κόσμο του λογισμικού, έχουν αναπτυχθεί πολλά εργαλεία που εξειδικεύονται σε αυτό το σκοπό. Μολονότι τα εργαλεία αυτά παριστάνουν τα δομικά στοιχεία του API, η εννοιολογία (semantics) παραμένει δύσκολο να εξαχθεί εντελώς αυτόματα. Κάποια από τα πιο δημοφιλή εργαλεία, είναι:

- Swaggerhub: Το SwaggerHub [11] είναι ένα online εργαλείο που σχεδιάστηκε για

να βοηθήσει τον σχεδιασμό, τη δημιουργία, την τεκμηρίωση και τη διαχείριση των RESTful APIs χρησιμοποιώντας το OpenAPI Specification [**?**]. Προσφέρει μια κεντρική πλατφόρμα όπου οι ομάδες μπορούν να συνεργαστούν σε όλη τη διάρκεια του κύκλου ζωής του API, από τον σχεδιασμό μέχρι την παραγωγή.

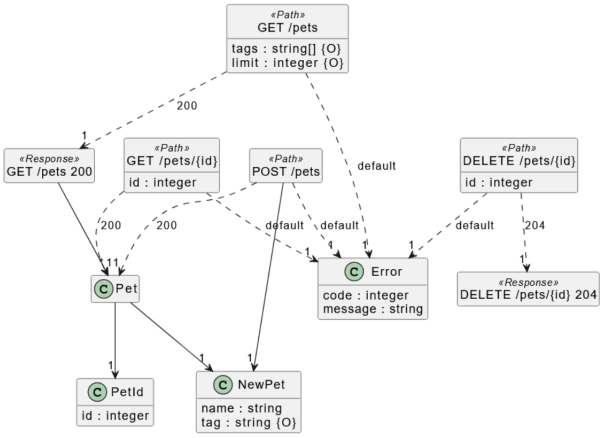
- Stoplight: Το Stoplight [12] είναι μια πλατφόρμα που προσφέρει εργαλεία και λύσεις για το σχεδιασμό, την ανάπτυξη, την τεκμηρίωση και τη δοκιμή API. Στοχεύει να απλοποιήσει την διαδικασία εργασίας των προγραμματιστών και των ομάδων ανάπτυξης λογισμικού, παρέχοντας ένα ενιαίο περιβάλλον για όλες τις φάσεις ανάπτυξης API.

- Redocly: Το Redocly [13] είναι μία πλατφόρμα που προσφέρει λύσεις για τη διαχείριση, δημιουργία, και δημοσίευση τεκμηρίωσης API. Η πλατφόρμα εστιάζει στη βελτίωση της εμπειρίας τόσο των προγραμματιστών που χρησιμοποιούν τα API όσο και των ομάδων που τα αναπτύσσουν, με σκοπό την απλοποίηση και την αυτοματοποίηση της διαδικασίας διαχείρισης API.

2.4.2 Εργαλεία για την παρουσίαση του API Documentation

Αντίστοιχα υπάρχουν εργαλεία για την παρουσίαση των REST APIs. Ένα documentation αποκτάει μεγαλύτερη αξία όταν μπορεί να παρουσιαστεί και μέσω ενός διαγράμματος. Κάποια από τα πιο δημοφιλή εργαλεία είναι:

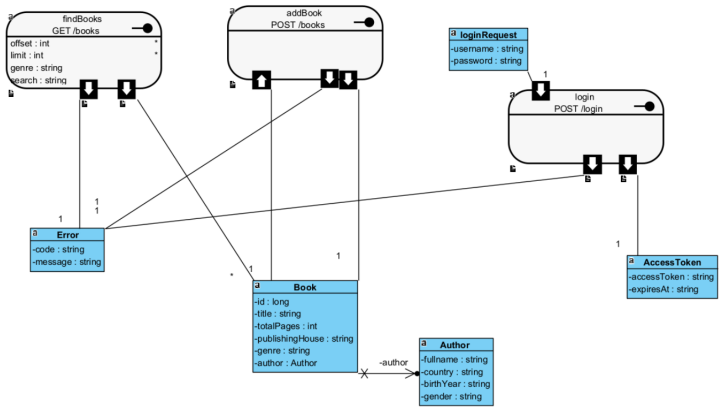
- Plant UML: Το PlantUML [14] είναι ένα open-source εργαλείο που χρησιμοποιείται για τη δημιουργία διαγραμμάτων UML [15] μέσω μιας απλής γλώσσας βασισμένης σε κείμενο. Επιτρέπει στους χρήστες να γράφουν περιγραφές διαγραμμάτων σε απλό κείμενο, το οποίο μετατρέπεται αυτόματα σε διαγράμματα. Αυτό καθιστά τη δημιουργία και την τροποποίηση διαγραμμάτων UML γρήγορη και απλή, καθώς δεν απαιτείται η χρήση περίπλοκων γραφικών διεπαφών.



Εικόνα 2.13: *Παράδειγμα παρουσίασης REST API μέσω του PlantUML*

- Visual Paradigm: Το Visual Paradigm [16] είναι ένα ισχυρό εργαλείο σχεδίασης και διαχείρισης για συστήματα Information Technology. Αποτελεί ένα εργαλείο UML CASE

που υποστηρίζει το UML 2, το SysML και το Business Process Modeling Notation. Επιτρέπει τη δημιουργία πλούσιων διαγραμμάτων UML διαφόρων τύπων όπως Class Diagrams [17] και Use Case diagrams [18]. Επίσης επιτρέπει την αυτόματη παραγωγή OpenAPI documentations για REST APIs που έχουν σχεδιαστεί στο εργαλείο αυτό.



Εικόνα 2.14: *Παράδειγμα παρουσίασης REST API μέσω του Visual Paradigm*

2.4.3 Εργαλείο Παραγωγής Εξαρτήσεων: Dependency Analyzer

Πολλές μελέτες έχουν πραγματοποιηθεί σχετικά με την ανάλυση των εξαρτήσεων των endpoints ενός API. Σε προηγούμενες μελέτες, ο Zhong Hao [19] και η Antonia Bertolino [20] επικεντρώθηκαν στους κανόνες παραμέτρων API και στα συμπεριφορικά μοντέλα για τη χρήση των web υπηρεσιών. Η μελέτη του Zhong Hao κατέταξε τους κανόνες παραμέτρων σε έξι κατηγορίες αλλά δεν διερεύνησε εκτενώς τις εξαρτήσεις μεταξύ των endpoints. Αυτοί οι κανόνες επικεντρώνονται στον τύπο της τιμής και χωρίζονται σε 6 κατηγορίες, δηλαδή: «Null», «Range», «Value», «Format», «Relation», «Other». Η Antonia Bertolino παρουσίασε ένα συμπεριφορικό μοντέλο με την ονομασία «Strawberry» αλλά επίσης δεν αντιμετώπισε τις εξαρτήσεις μεταξύ τερματικών σημείων. Ο Oostvogels N [21] και ο Martin-Lopez A [22] εξέτασαν τις εξαρτήσεις μεταξύ παραμέτρων στα APIs, επισημαίνοντας διάφορους τύπους εξαρτήσεων, δηλαδή «Requires», «Or», «OnlyOne», «AllOrNone», «ZeroOrOne», «Arithmetic/Relational», «Complex». Ωστόσο, αυτές αφορούν κυρίως παραμέτρους εντός του endpoint και δεν αντιμετωπίζουν εξαρτήσεις μεταξύ διαφορετικών endpoint. Ο Xiaoying Bai [23] και ο Animesh Chaturvedi [24] επικεντρώθηκαν στις εξαρτήσεις μεταξύ διαφορετικών λειτουργιών, ιδιαίτερα στη σειρά με την οποία πρέπει να εκκληθούν οι λειτουργίες για τη σωστή λειτουργικότητα της υπηρεσίας. Περιέγραψαν διάφορους τύπους εξαρτήσεων και χρησιμοποίησαν ανάλυση αλλαγής μεταξύ λειτουργιών, με στόχο τη δημιουργία περιπτώσεων δοκιμής. Η εμβέλειά τους όμως ήταν πιο περιορισμένη και επικεντρωμένη στη δημιουργία περιπτώσεων δοκιμής.

Στα πλαίσια προηγούμενης διπλωματικής εργασίας του Softlab [1], υλοποιήθηκε ένα ολοκληρωμένο εργαλείο ανάλυσης και παραγωγής εξαρτήσεων ενός REST API που ονομάζεται Dependency Analyzer. Το Dependency Analyzer περιλαμβάνει μια ευρύτερη εξέταση των παραμέτρων API και των εξαρτήσεων μεταξύ των endpoints. Το σύστημα αυτό αποτελεί-

ται από ένα Python Script, το οποίο δέχεται ως όρισμα ένα Postman Collection αρχείο και παράγει ένα .txt αρχείο το οποίο περιέχει τις παραγόμενες εξαρτήσεις μεταξύ των end-points. Στην τρέχουσα διπλωματική εργασία, θα αξιοποιήσουμε το Dependency Analyzer και τις δυνατότητες που προσφέρει με σκοπό την κατασκευή ενός εργαλείου το οποίο θα οπτικοποιεί αυτές τις εξαρτήσεις σε μορφή γράφου.

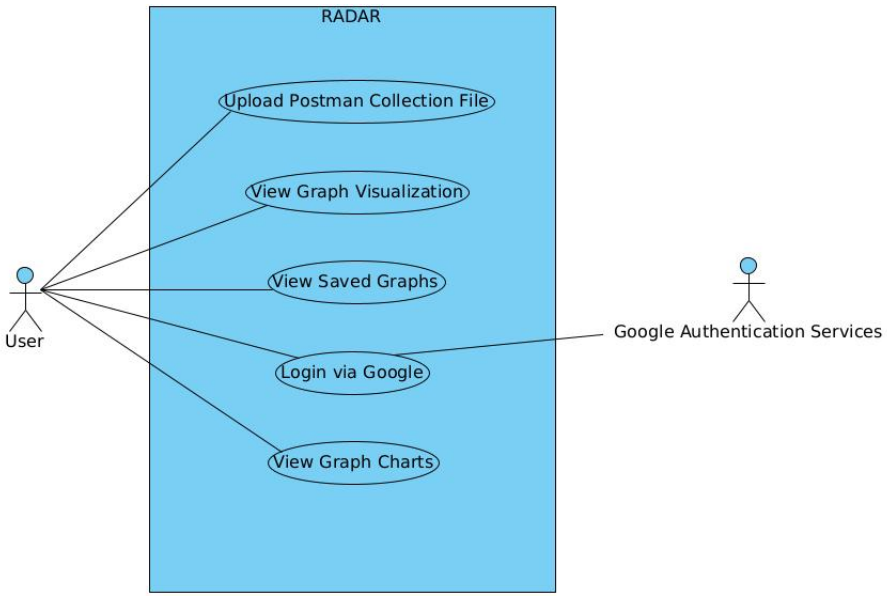
Κεφάλαιο 3

RADAR: Οπτικοποίηση εξαρτήσεων μεταξύ API endpoints

3.1 Απαιτήσεις

Πολλές φορές η σχέση εξάρτησης μεταξύ των endpoints ενός API, όπως αναφέραμε και προηγουμένως, δεν είναι εμφανής. Η εξαγωγή των εξαρτήσεων με μη αυτοματοποιημένο τρόπο σε συστήματα όπως για παράδειγμα η Paypal, τα οποία αποτελούνται από δεκάδες αν όχι εκατοντάδες endpoints, θα απαιτούσε πολλές εργατοώρες, ενώ δε θα αποτελούσε μια βιώσιμη λύση, καθώς η προσθήκη ενός endpoint θα μπορούσε να επηρεάσει ή και να αλλάξει τις προϋπάρχουσες εξαρτήσεις. Το σύστημα το οποίο υλοποιήθηκε στα πλαίσια της παρούσας διπλωματικής ως δικτυακή υπηρεσία έχει ως σκοπό την αυτόματη παραγωγή και οπτικοποίηση αυτών των εξαρτήσεων.

3.2 Περιπτώσεις Χρήσεις - Λειτουργία Εφαρμογής



Εικόνα 3.1: *Use case diagram of RADAR*

3.2.1 Περίπτωση Χρήσης 1: Ανέβασμα Αρχείου και Εμφάνιση Γράφου Εξαρτήσεων

Ο χρήστης μπαίνει στην web εφαρμογή και συμπληρώνει την φόρμα υποβολής του αρχείου. Μόλις πατήσει το κουμπί του Upload, το αρχείο στέλνεται στο backend και επιστρέφονται οι παραγόμενες εξαρτήσεις. Το frontend επεξεργάζεται την απάντηση που έλαβε από το backend, παράγει τον γράφο οπτικοποίησης και τον εμφανίζει στο χρήστη. Ο χρήστης μπορεί να αλληλεπιδράσει με το γράφο με τους παρακάτω τρόπους:

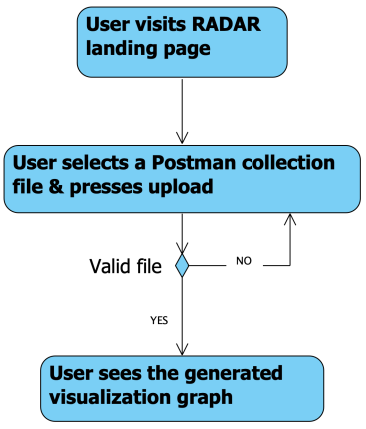
1. Hover πάνω σε κόμβο: Κάνοντας hover πάνω σε έναν κόμβο, εμφανίζονται όλα τα dependencies του κόμβου. Οι ακμές οι οποίες φεύγουν από τον κόμβο τονίζονται με μπλε χρώμα, ενώ οι ακμές που εισέρχονται στον κόμβο με κόκκινο.

2. Κλικ πάνω σε κόμβο: Επιλέγοντας έναν κόμβο, εμφανίζεται στα δεξιά της οθόνης μία λίστα με όλα τα dependencies του επιλεγμένου κόμβου. Ακολουθώντας τη λογική του swagger documentation, ο χρήστης μπορεί να κάνει expand κάθε στοιχείο της λίστας και να δει το mapping των attributes από τα οποία προκύπτει το dependency, καθώς και κάποιες πληροφορίες για αυτά τα attributes, όπως για παράδειγμα το data type του attribute ή το που εμφανίζεται το attribute. Επιπλέον, στο πάνω μέρος της οθόνης υπάρχουν κάποια controls για το γράφο. Όταν υπάρχει επιλεγμένο κάποιο node, ο χρήστης μπορεί να κάνει isolate το node και όλα τα dependencies. Αυτή η λειτουργία είναι χρήσιμη σε περιπτώσεις γράφων με πάρα πολλά nodes και edges όπου θέλουμε να επικεντρωθούμε σε κάποιο συγκεκριμένο κόμβο.

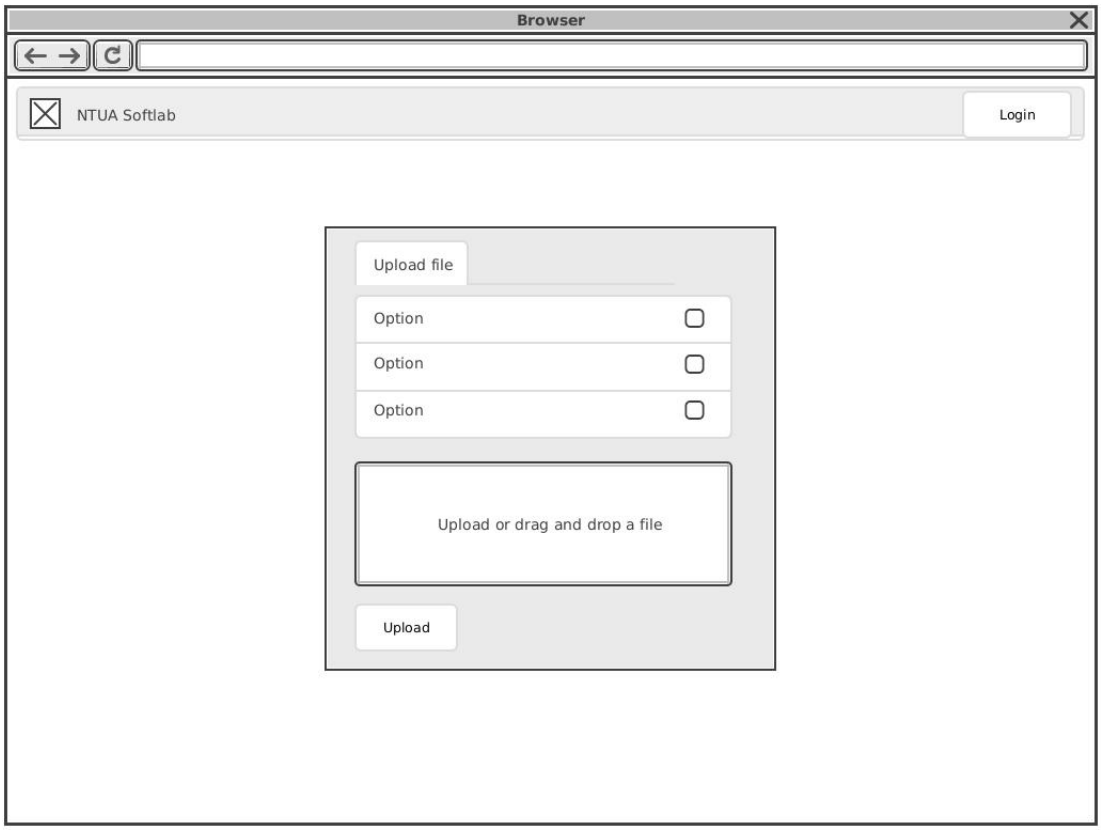
3. Show graph info: Ο χρήστης μπορεί να ενεργοποιήσει το switch που βρίσκεται πάνω αριστερά στη σελίδα και να δει κάποιες πληροφορίες σχετικά με τον παραγόμενο γράφο.

4. Μετακίνηση nodes: Ο χρήστης μπορεί προσωρινά να σύρει και να τοποθετήσει τα nodes του γράφου σε όποιο σημείο επιθυμεί.

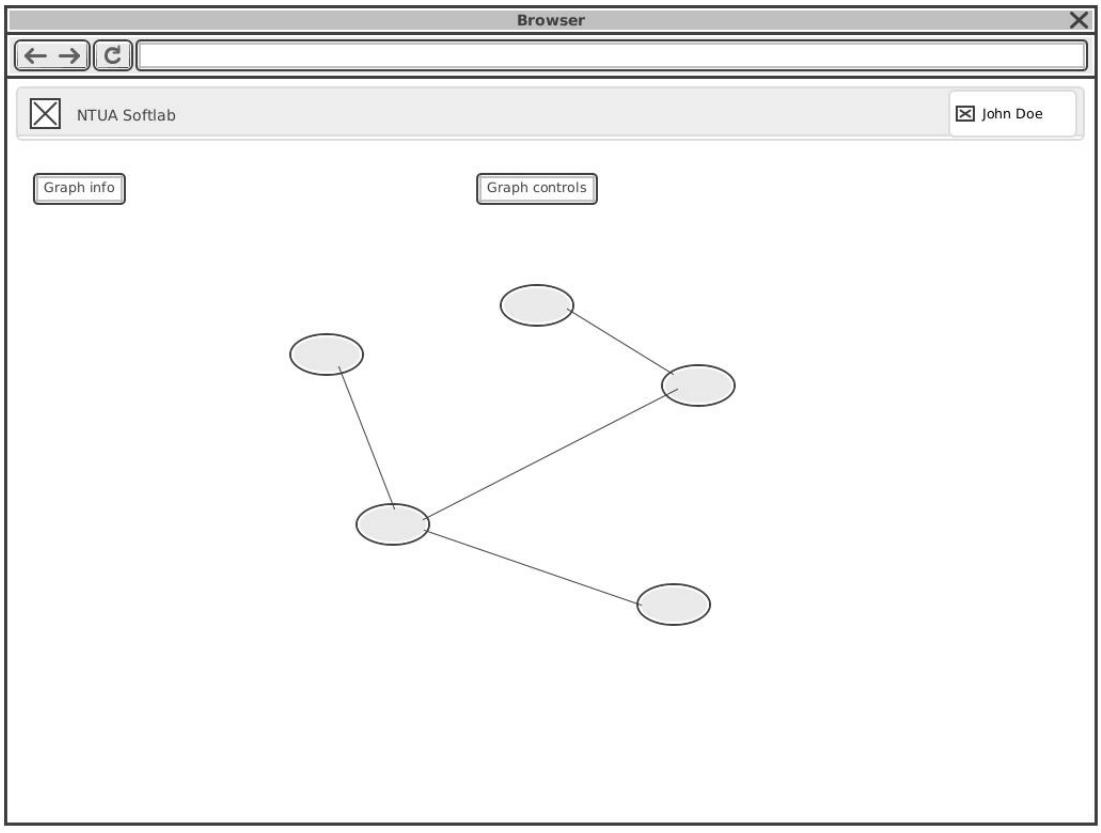
5. Show nodes χωρίς dependencies: Ο χρήστης μπορεί από τα graph controls στο πάνω μέρος της σελίδας να εμφανίσει κόμβους οι οποίοι δεν έχουν κανένα dependency.



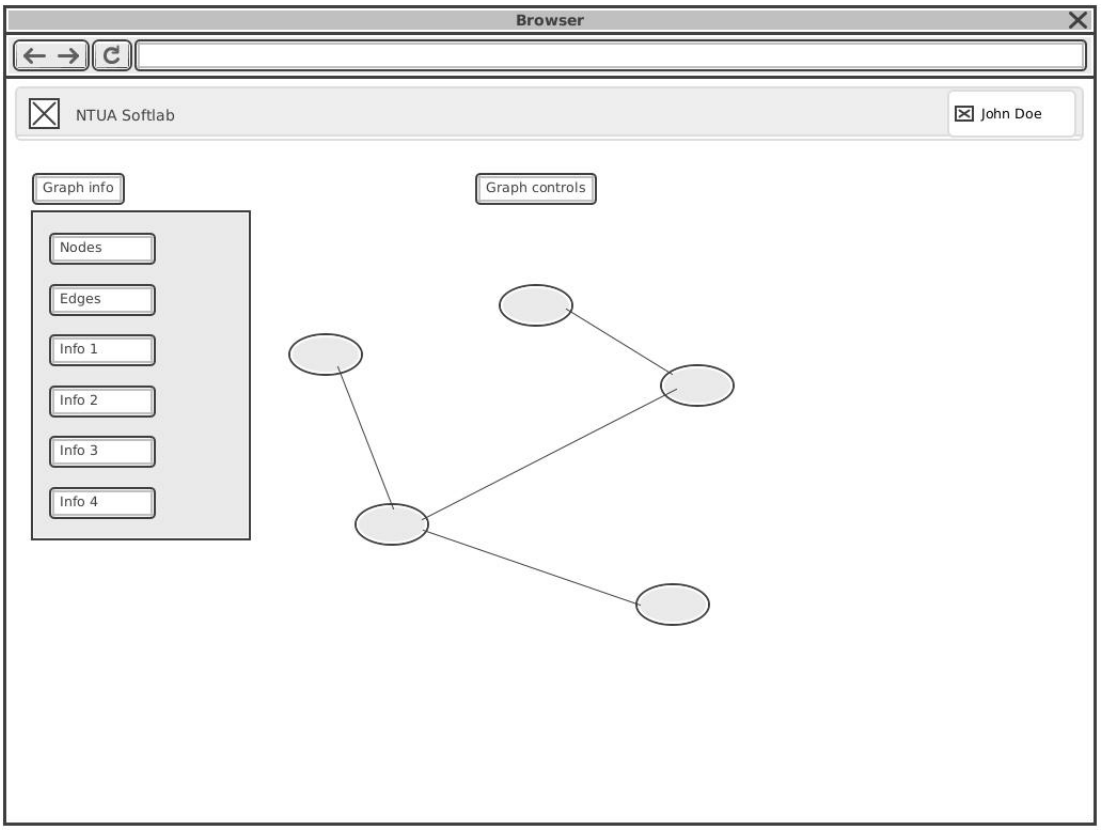
Εικόνα 3.2: *Activity Diagram: Use Case 1*



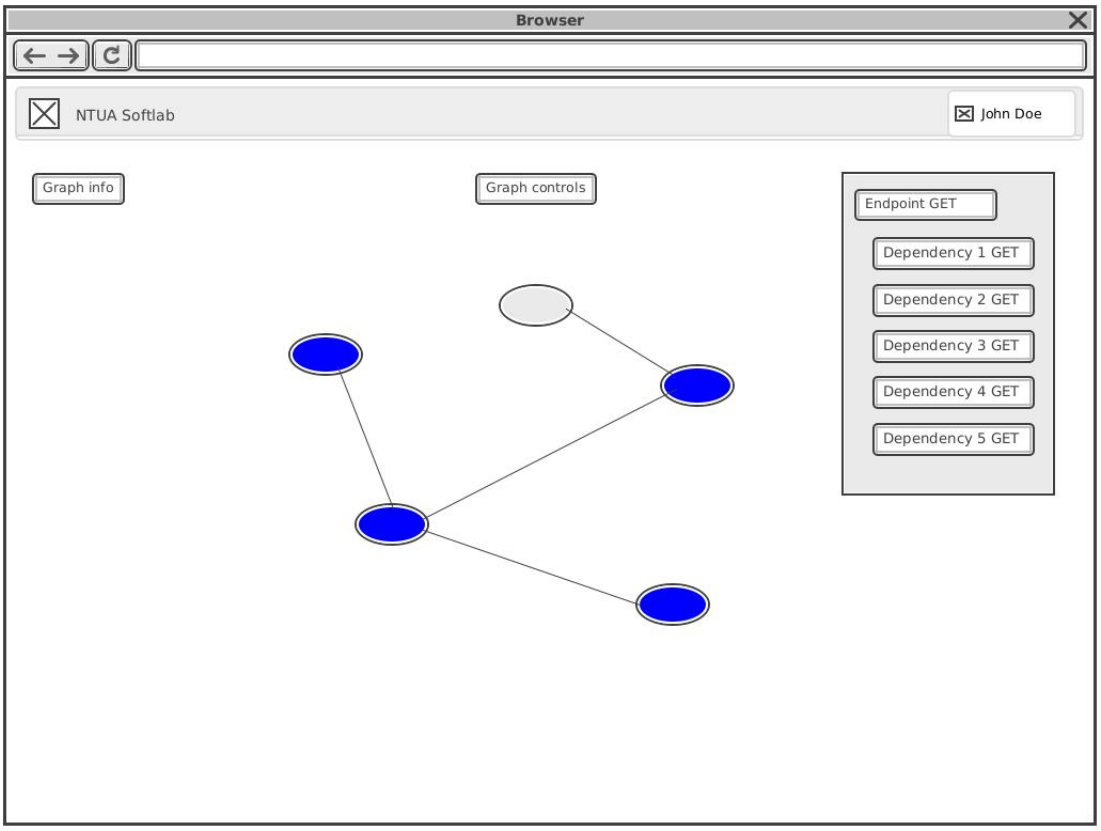
Εικόνα 3.3: *Upload Postman Collection Wireframe*



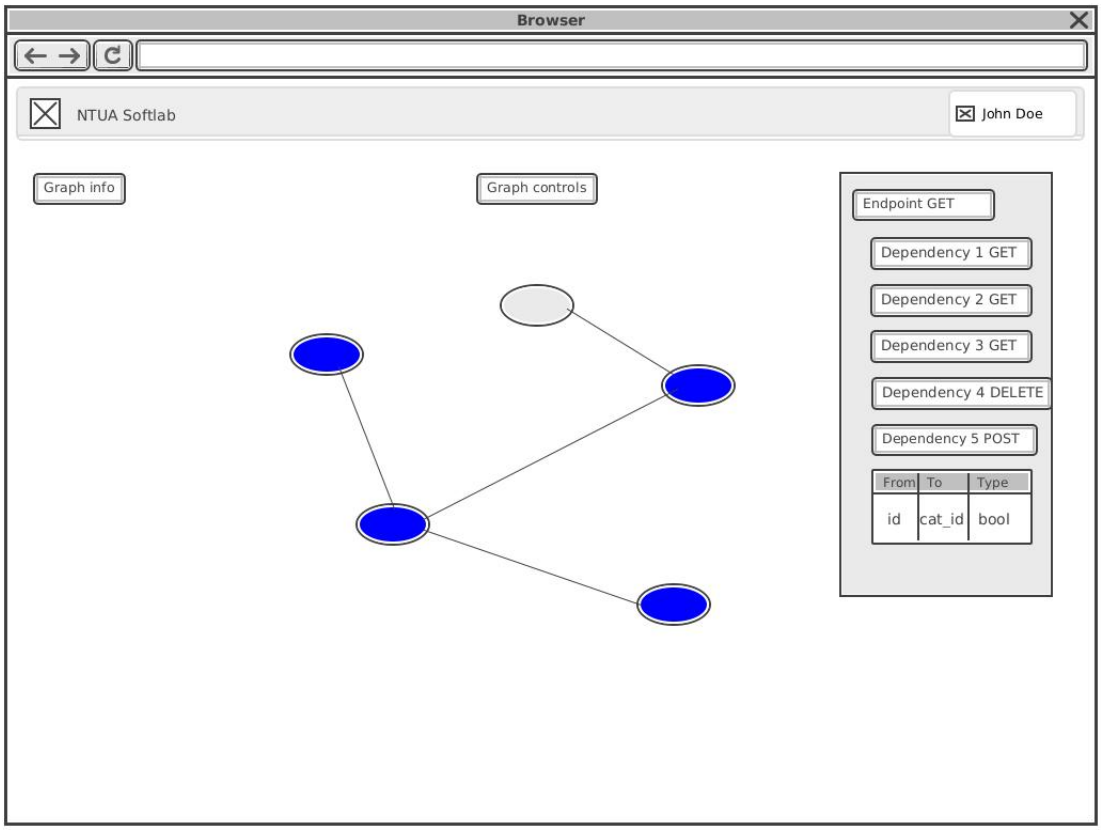
Εικόνα 3.4: *Graph visualization Wireframe*



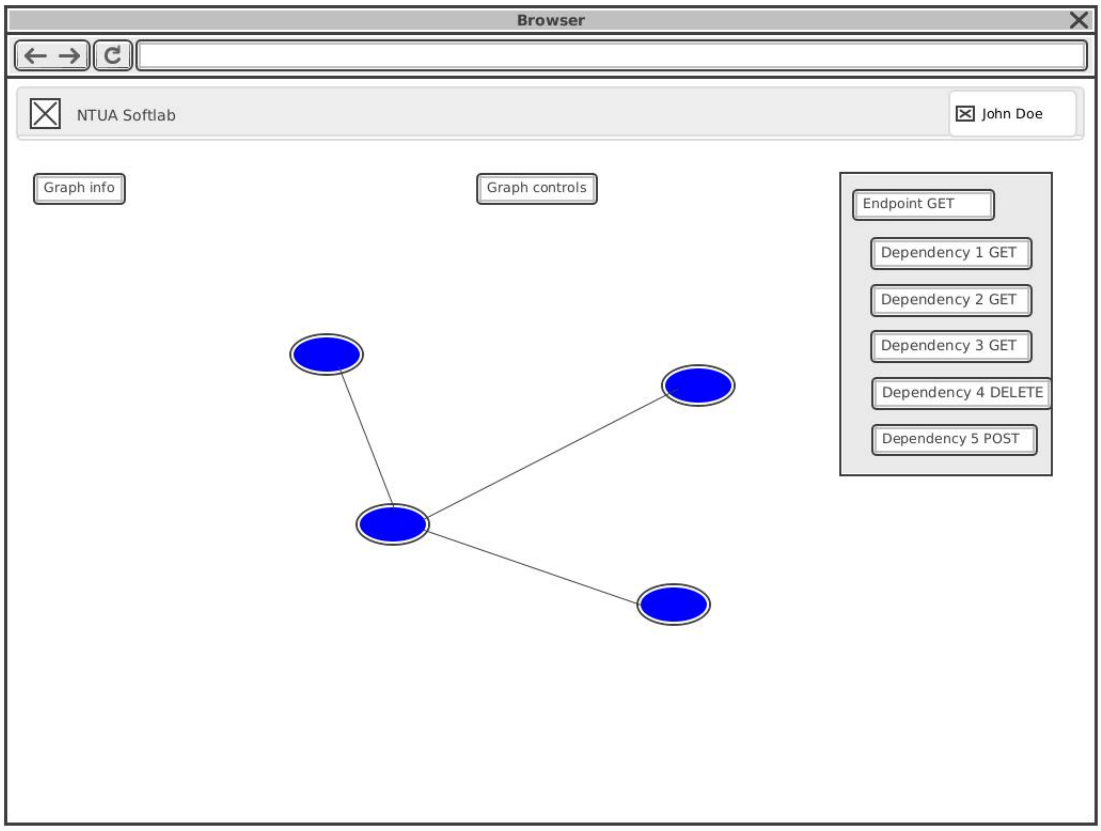
Εικόνα 3.5: *Graph visualization info Wireframe*



Εικόνα 3.6: *Graph visualization Selected Node Wireframe*



Εικόνα 3.7: *Graph visualization Selected Node Expanded Wireframe*

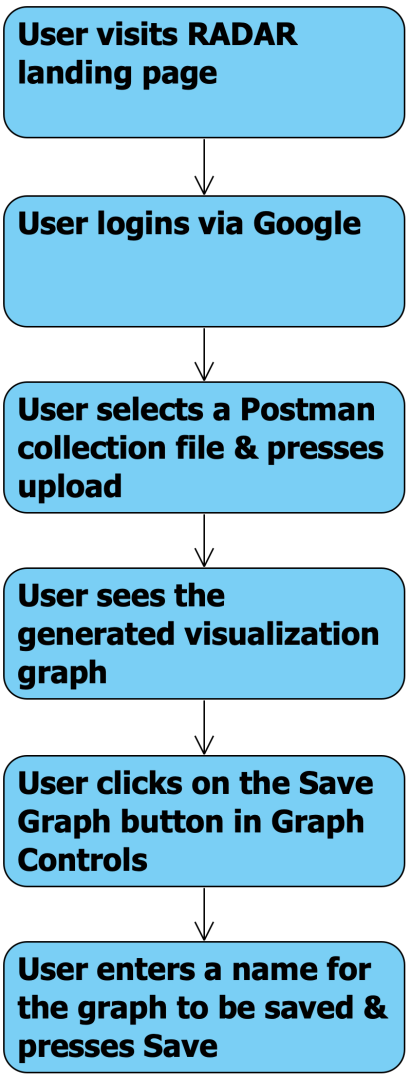


Εικόνα 3.8: *Graph visualization Isolated Node Wireframe*

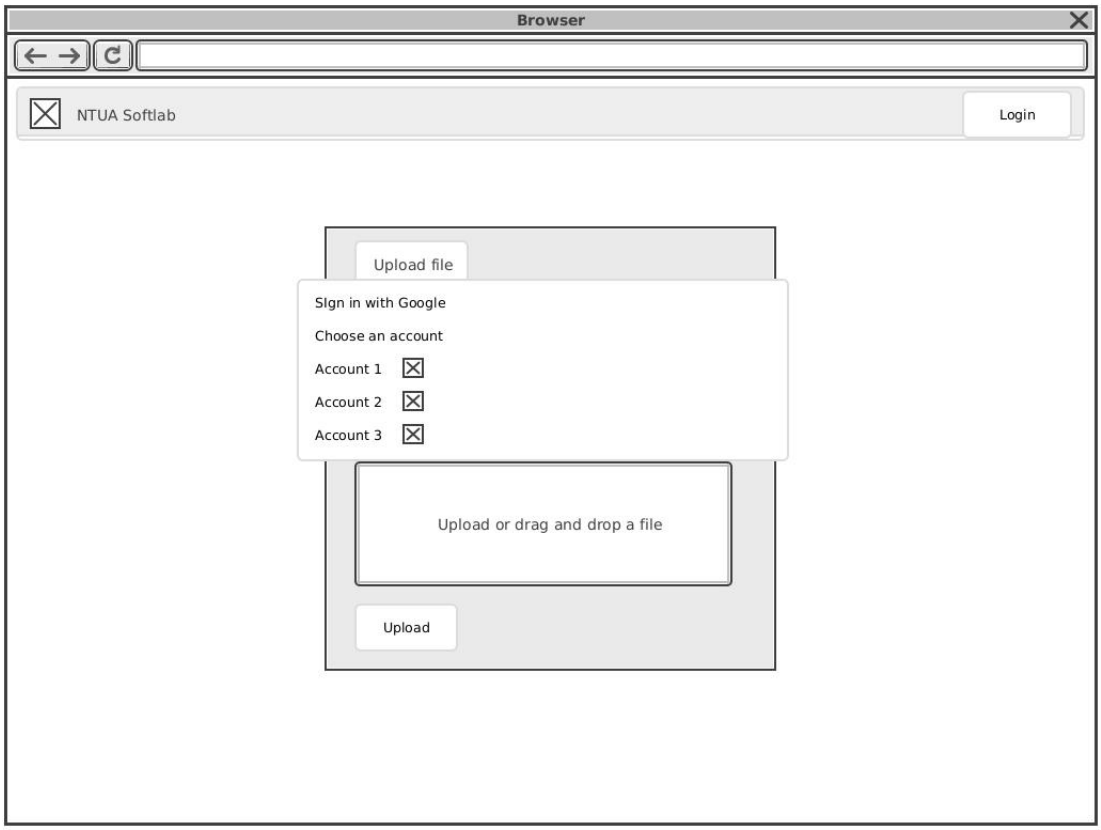
3.2.2 Περίπτωση Χρήσης 2: Αποθήκευση Γράφου για μελλοντική χρήση

Ο χρήστης μπαίνει στην web εφαρμογή και πατάει το κουμπί Sign in via google. Μόλις γίνει authenticated, συμπληρώνει και πάλι την φόρμα υποβολής του αρχείου και πατάει Upload. Ο χρήστης μπορεί να αλληπιδράσει με το γράφο όπως περιγράψαμε στην Περίπτωση Χρήσης 1, ενώ επίσης:

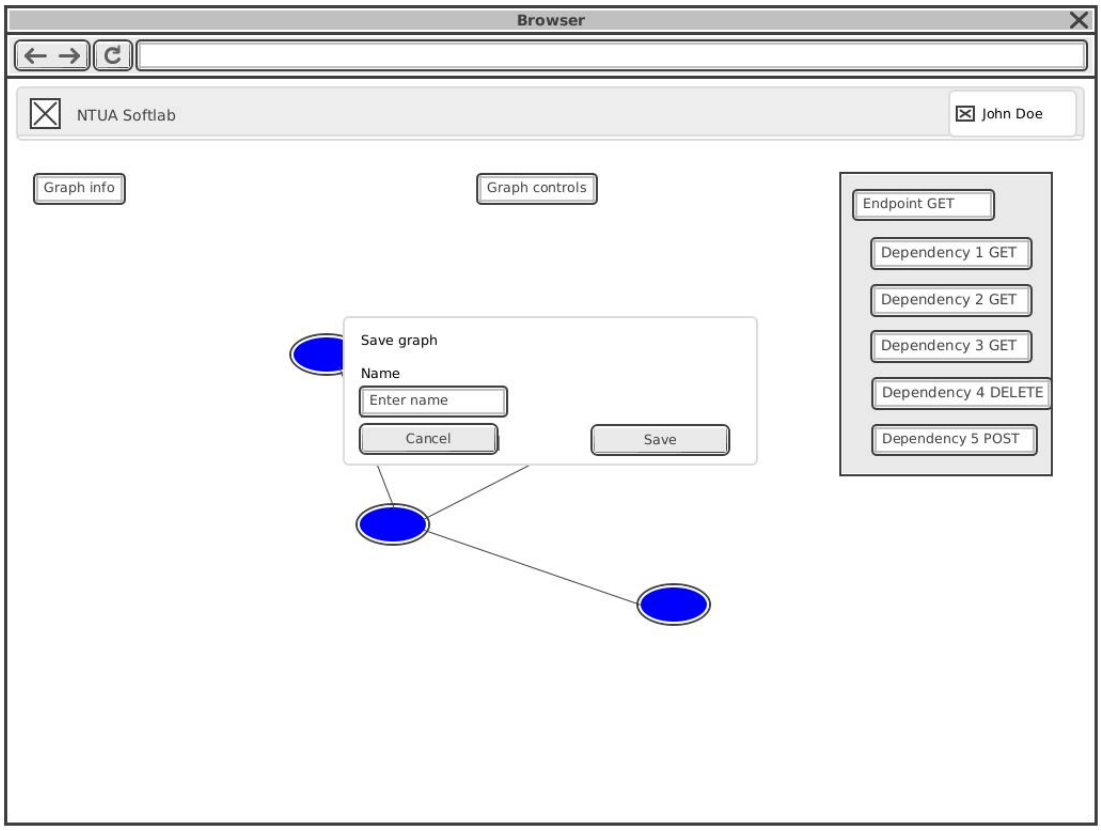
- Πατώντας από τα graph controls στο πάνω μέρος της σελίδας το Save Graph icon, εμφανίζεται ένα popup, στο οποίο ο χρήστης μπορεί να ονομάσει όπως θέλει τον γράφο που θέλει να αποθηκεύσει. Το default όνομα που του δίνεται και είναι προεπιλεγμένο στο name input, είναι το όνομα του αρχείου. Μόλις ο χρήστης επιλέξει το όνομα που επιθυμεί και πατήσει Save, ο γράφος αποθηκεύεται στη βάση. Ο χρήστης ενημερώνεται με κατάλληλο μήνυμα συστήματος έαν η ενέργεια του ήταν επιτυχής ή όχι.



Εικόνα 3.9: *Activity Diagram: Use Case 2*



Εικόνα 3.10: *Login with Google Wireframe*



Εικόνα 3.11: *Graph visualization Save Modal Wireframe*

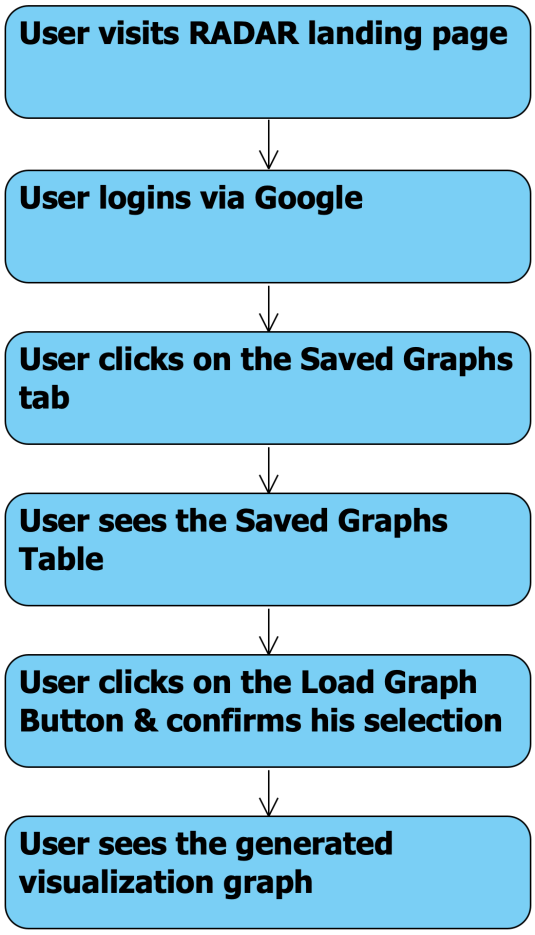
3.2.3 Περίπτωση Χρήσης 3: Εμφάνιση γράφου χωρίς ανέβασμα αρχείου

Ο χρήστης μπαίνει στην web εφαρμογή και πατάει το κουμπί Sign in via google. Μόλις γίνει authenticated, ενεργοποείται το Saved graphs tab. Ο χρήστης πατώντας σε αυτό το tab, μπορεί να δει όλους τους γράφους τους οποίους είχε αποθηκεύσει στο παρελθόν

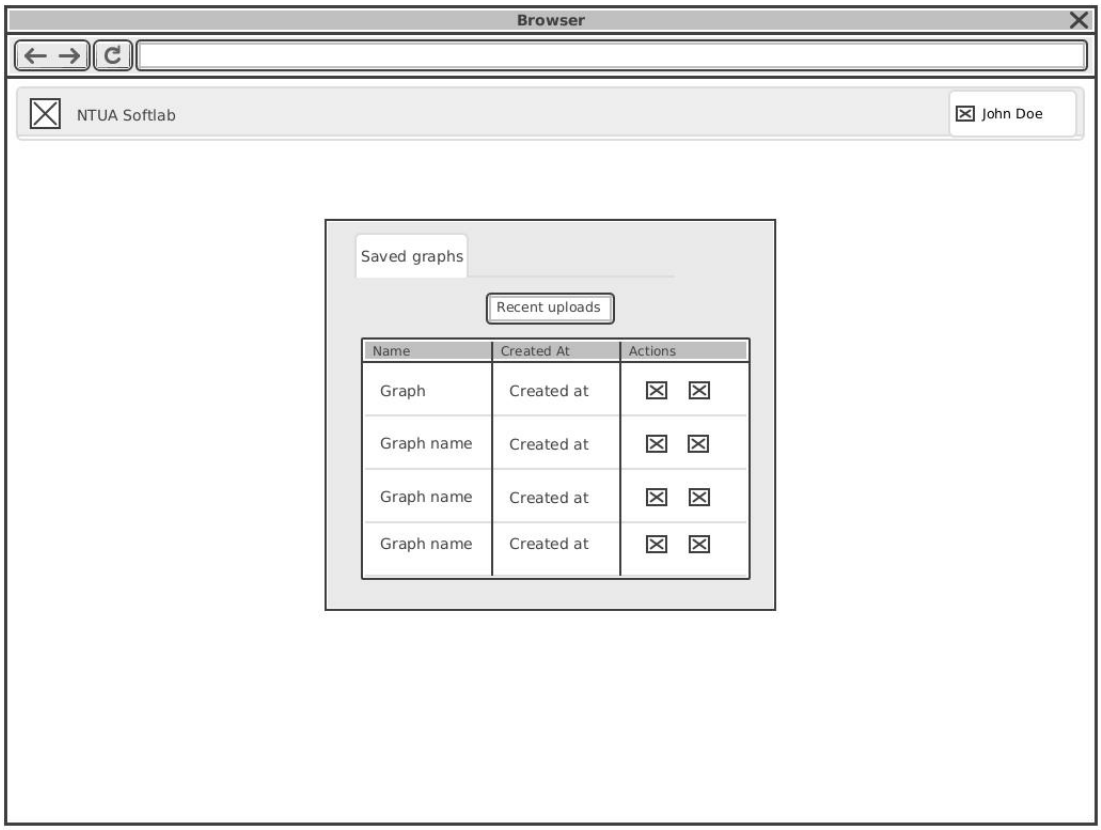
- Πατώντας στο Load Graph Icon, εμφανίζεται ο γράφος οπτικοποίησης στο χρήστη, με τον οποίο μπορεί να αλληλεπιδράσει ακριβώς όπως περιγράφουμε στις Περιπτώσεις 1 και 2.

- Ο χρήστης μπορεί επίσης έαν επιθυμεί να διαγράψει κάποιον αποθηκευμένο γράφο πατώντας το Delete Graph Icon. Εμφανίζεται ένα popup επιβεβαίωσης και με βάση την επιλογή του χρήστη, ο γράφος διαγράφεται από τη βάση.

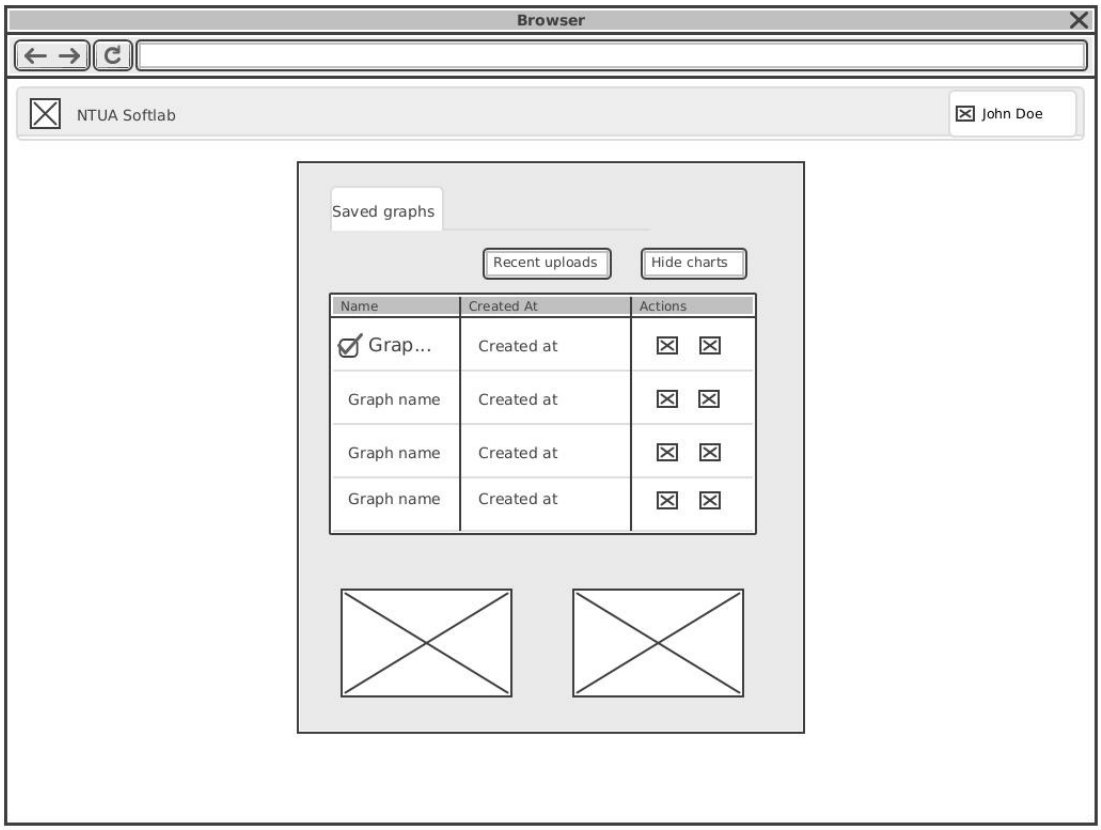
- Ο χρήστης μπορεί στο Saved Graphs Tab, να επιλέξει πολλαπλούς αποθηκευμένους γράφους και να πατήσει το Show Charts κουμπί που εμφανίζεται στο πάνω δεξιά μέρος του tab. Πατώντας αυτό το κουμπί, εμφανίζονται κάποια διαγράμματα που περιέχουν πληροφορίες σχετικά με τους επιλεγμένους γράφους.



Εικόνα 3.12: *Activity Diagram: Use Case 3*

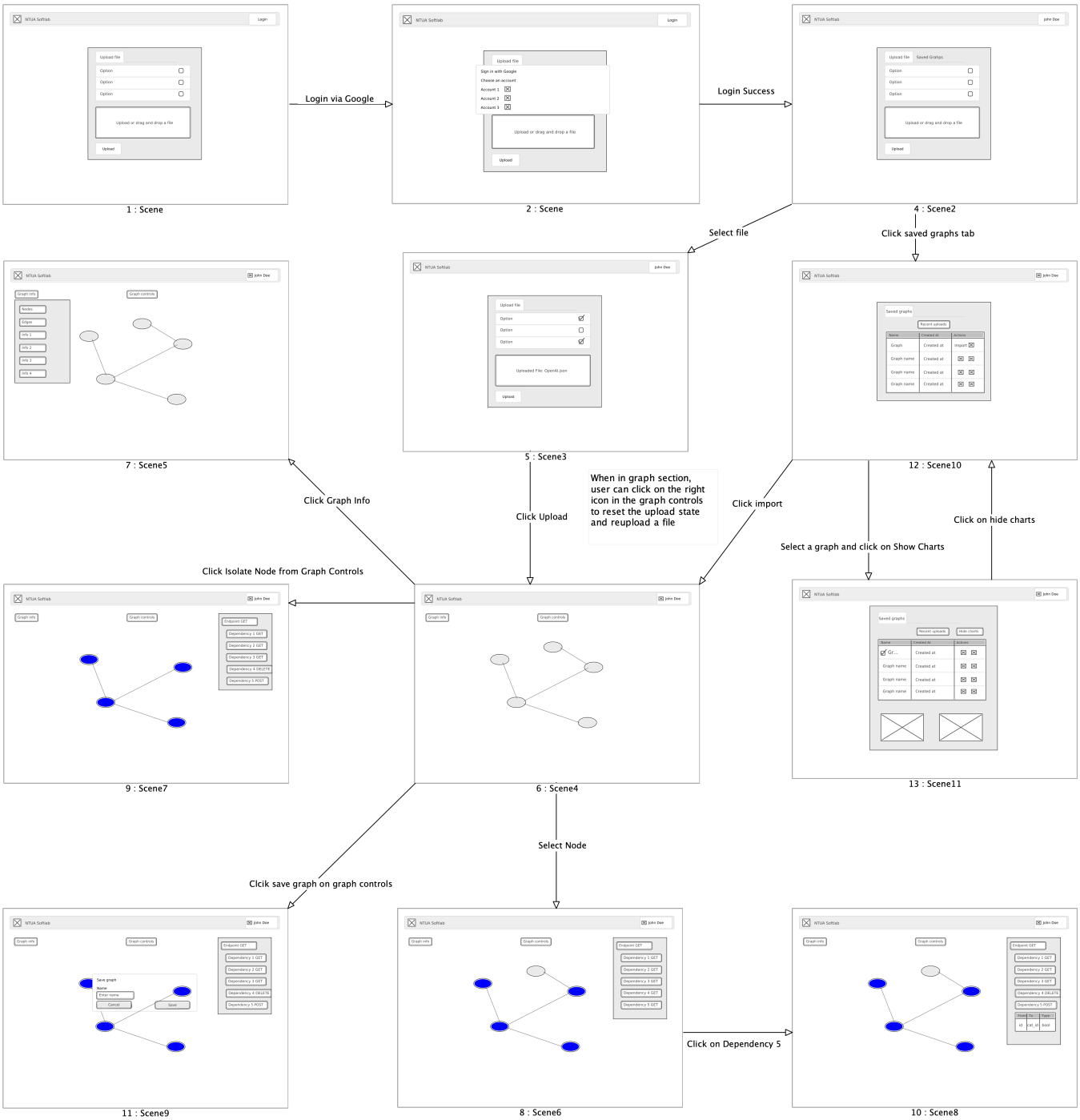


Εικόνα 3.13: *Saved graphs list Wireframe*



Εικόνα 3.14: *Saved graphs with charts Wireframe*

3.2.4 Wireflow

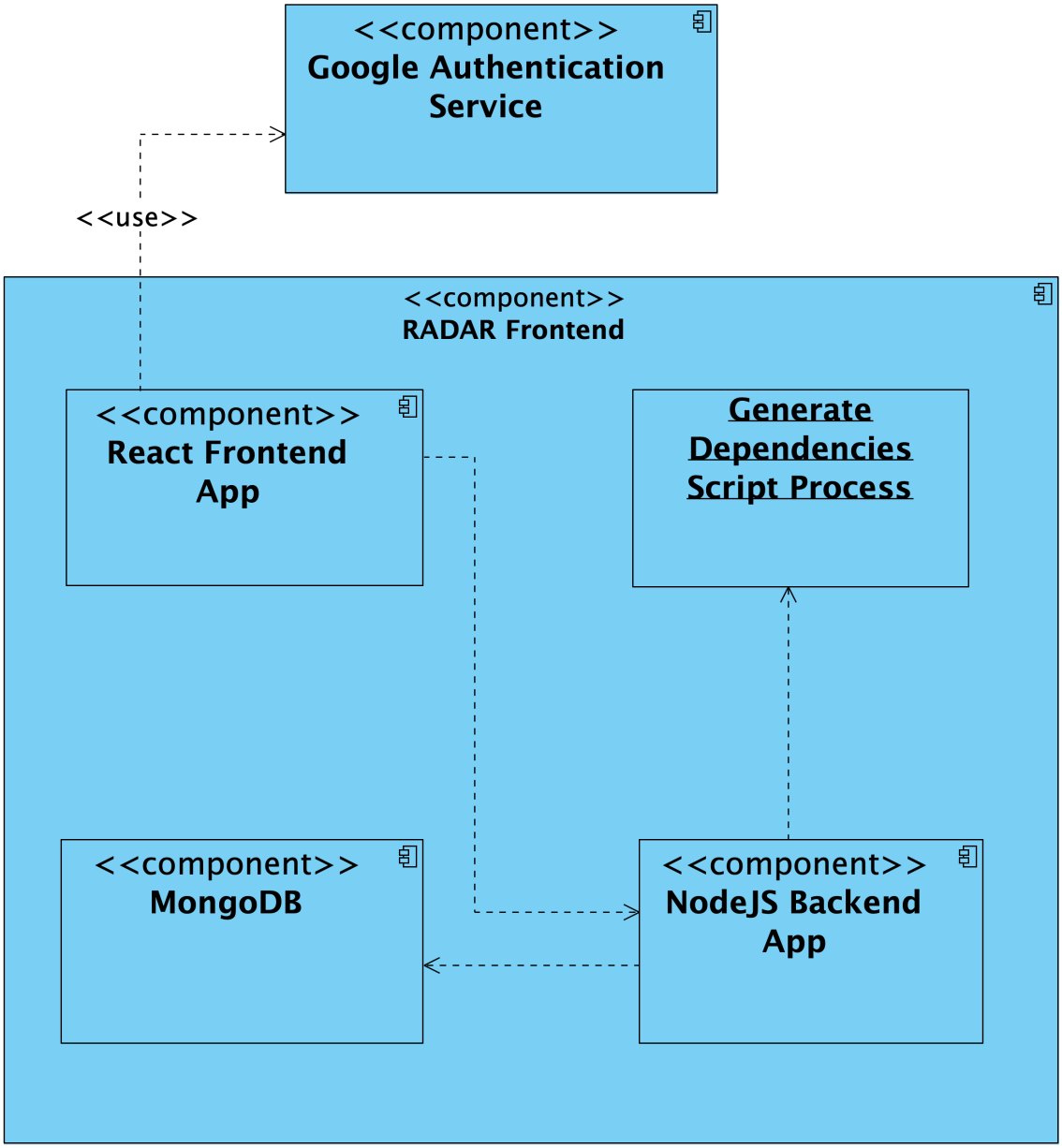


Εικόνα 3.15: *Wireflow Diagram*

3.3 Αρχιτεκτονική Εφαρμογής

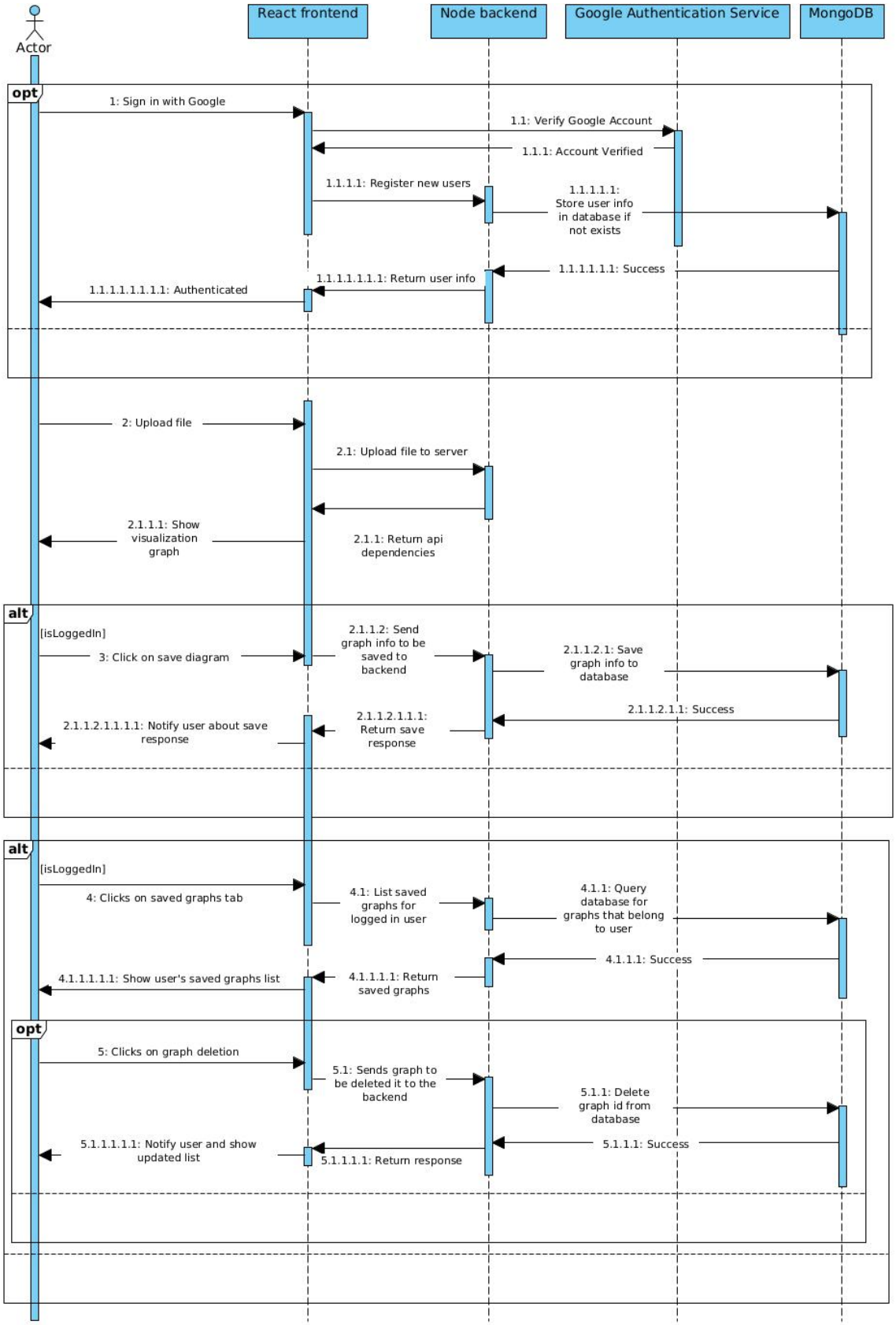
Το σύστημα που αναπτύξαμε ονομάζεται RADAR. Παρακάτω παρουσιάζονται τα απαιραίτητα component, sequence και deployment UML διαγράμματα τα οποία περιγράφουν την αρχιτεκτονική και τις λειτουργίες του συστήματος μας.

3.3.1 Component Diagram



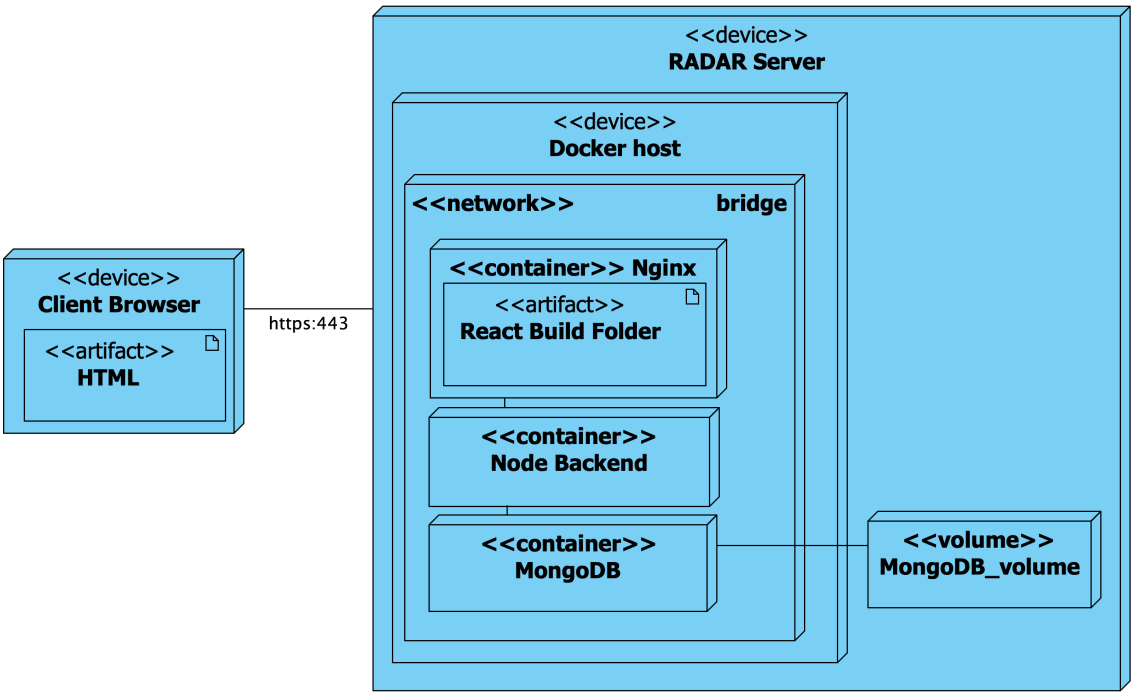
Εικόνα 3.16: *Component Diagram of RADAR*

3.3.2 Sequence Diagram



Εικόνα 3.17: *Sequence Diagram of RADAR*

3.3.3 Deployment Diagram



Εικόνα 3.18: *Deployment Diagram of RADAR*

3.4 Υλοποίηση

Η εφαρμογή μας αποτελείται από δύο συστήματα: το backend και το frontend τα οποία επικοινωνούν μεταξύ τους και εξασφαλίζουν τη λειτουργικότητα της εφαρμογής.

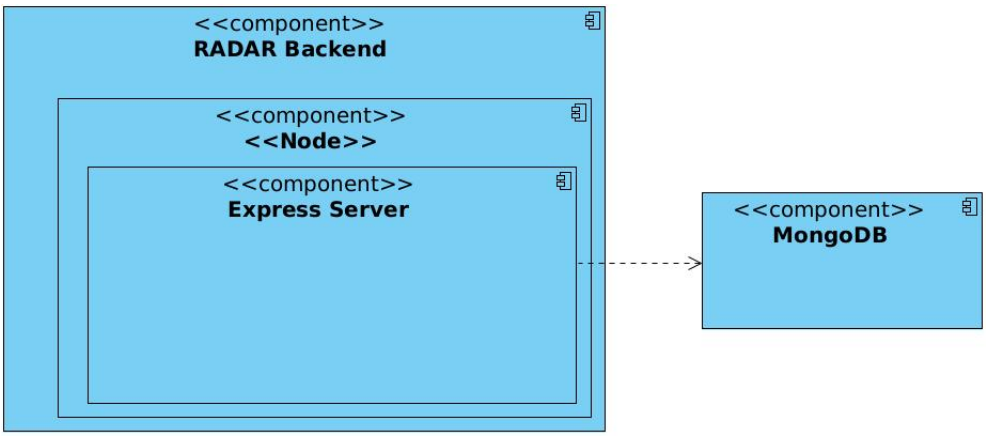
3.4.1 Backend

Το backend της εφαρμογής υλοποιήθηκε με τη χρήση των παρακάτω τεχνολογιών:

1. Node.js: Το Node.js [25] είναι ένα cross-platform, open-source JavaScript runtime environment, σχεδιασμένο να για την ανάπτυξη scaleable web εφαρμογών. Αυτό σημαίνει ότι είναι ιδανικό για την ανάπτυξη server-side λειτουργιών που απαιτούν υψηλή απόδοση και προσβασιμότητα, όπως APIs. Επιλέχθηκε το Node.js για την υλοποίηση του backend διότι προσφέρει ένα αποδοτικό και ελαφρύ περιβάλλον εκτέλεσης JavaScript, ικανό να διαχειρίζεται πολυάριθμες παράλληλες συνδέσεις με ελάχιστο κόστος. Η μη-αποκλειστική, βασισμένη σε γεγονότα αρχιτεκτονική του επιτρέπει την εύκολη δημιουργία επεκτάσιμων web εφαρμογών, καθιστώντας το ιδανικό για την ανάπτυξη του backend μέρους της εφαρμογής μου, όπου η απόδοση και η συνεχής διαθεσιμότητα είναι κρίσιμης σημασίας.

2. Express.js: Το Express.js [26] είναι ένα ελαφρύ και ευέλικτο Node.js web application framework που παρέχει ένα ισχυρό σύνολο λειτουργιών για web και mobile εφαρμογές. Διευκολύνει την ανάπτυξη web εφαρμογών μέσω της παροχής εργαλείων για τη δημιουργία api endpoints, τη διαχείριση αιτήσεων και αποκρίσεων, καθώς και τη χρήση middleware, κάνοντας πιο αποτελεσματική την ανάπτυξη του backend.

3. MongoDB: Η MongoDB [27] είναι μια NoSQL βάση δεδομένων που χρησιμοποιεί ένα μη σχεσιακό μοντέλο αποθήκευσης. Είναι σχεδιασμένη για την αποθήκευση δομημένων δεδομένων ως json documents, προσφέροντας τεράστια ευελιξία και κλιμάκωση. Τη χρησιμοποιήσαμε στο πρόγραμμα λόγω της δυνατότητάς της να χειρίζεται μεγάλες ποσότητες δεδομένων, της απλότητας στην ανάκτηση και ενημέρωση δεδομένων, και της ικανότητας να υποστηρίζει δυναμικές δομές δεδομένων χωρίς σταθερό schema.

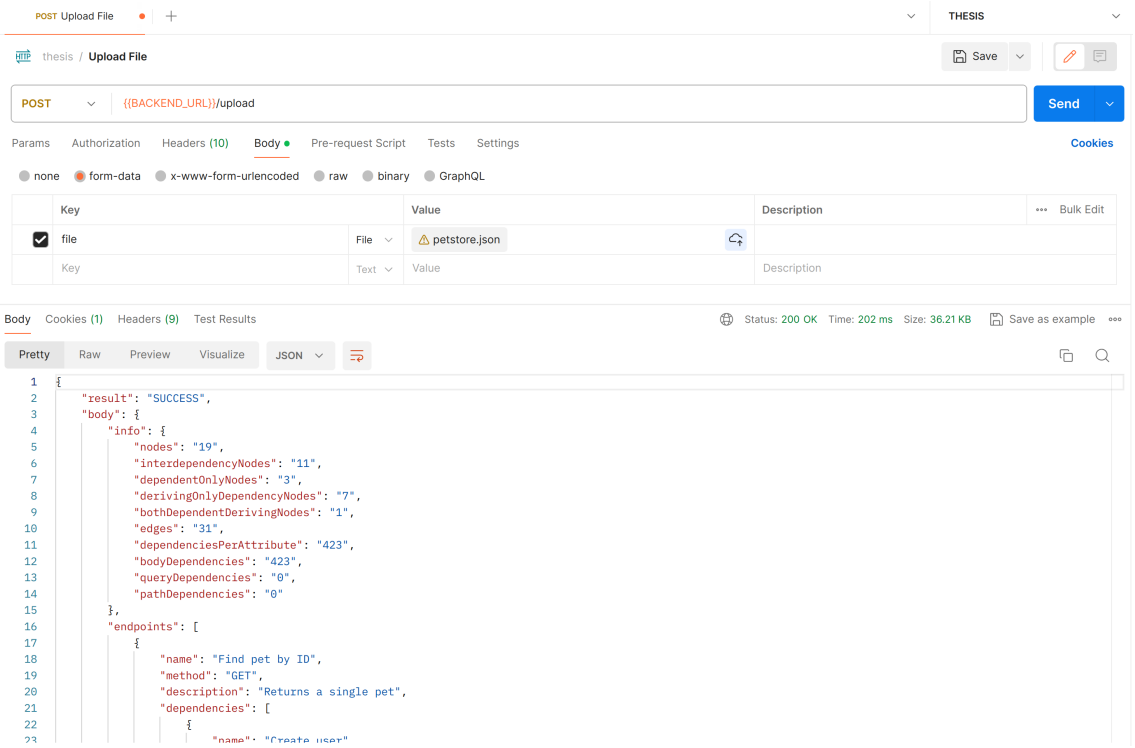


Εικόνα 3.19: *Component Diagram of RADAR Backend*

Με τη χρήση των προαναφερόμενων τεχνολογιών, σχεδιάσαμε ένα API, το οποίο είναι υπεύθυνο για την εξυπηρέτηση τριών βασικών λειτουργιών:

1. File Upload: Δημιουργήσαμε ένα endpoint το οποίο είναι υπεύθυνο για το ανέβασμα του αρχείου στον server.

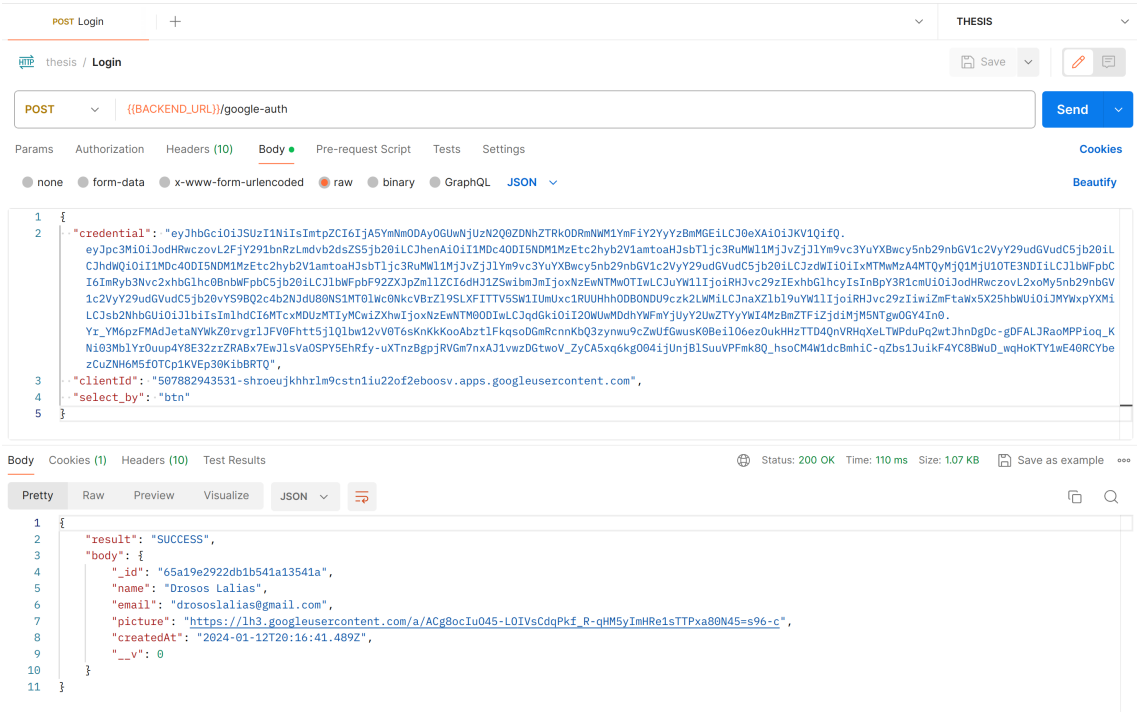
   (α') POST /api/upload: Το api διαβάζει το αρχείο που στέλνεται στο request body, εκτελεί το script παραγώγης εξαρτήσεων και επιστρέφει τις παραγόμενες εξαρτήσεις στο response.



Εικόνα 3.20: *Upload File Request Example*

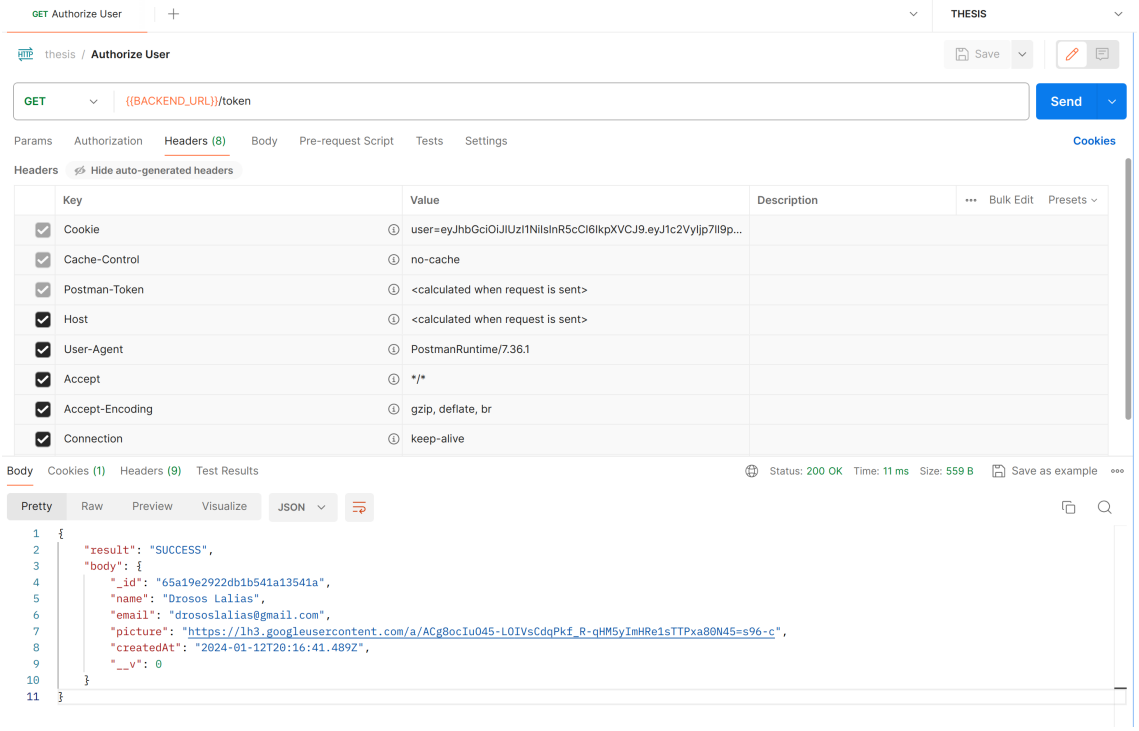
2. Authentication: Δημιουργήθηκαν δύο endpoints υπεύθυνα για το authentication του χρήστη.

   (α') POST /api/google-auth: Verify ό,τι ο χρήστης που προσπαθεί να κάνει login μέσω της google έχει valid google account. Έαν το verification είναι επιτυχές, αποθηκεύουμε τα απαραίτητα στοιχεία του χρήστη, όπως το όνομα του το email του, στη βάση δεδομένων μας.



Εικόνα 3.21: *Login Request Example*

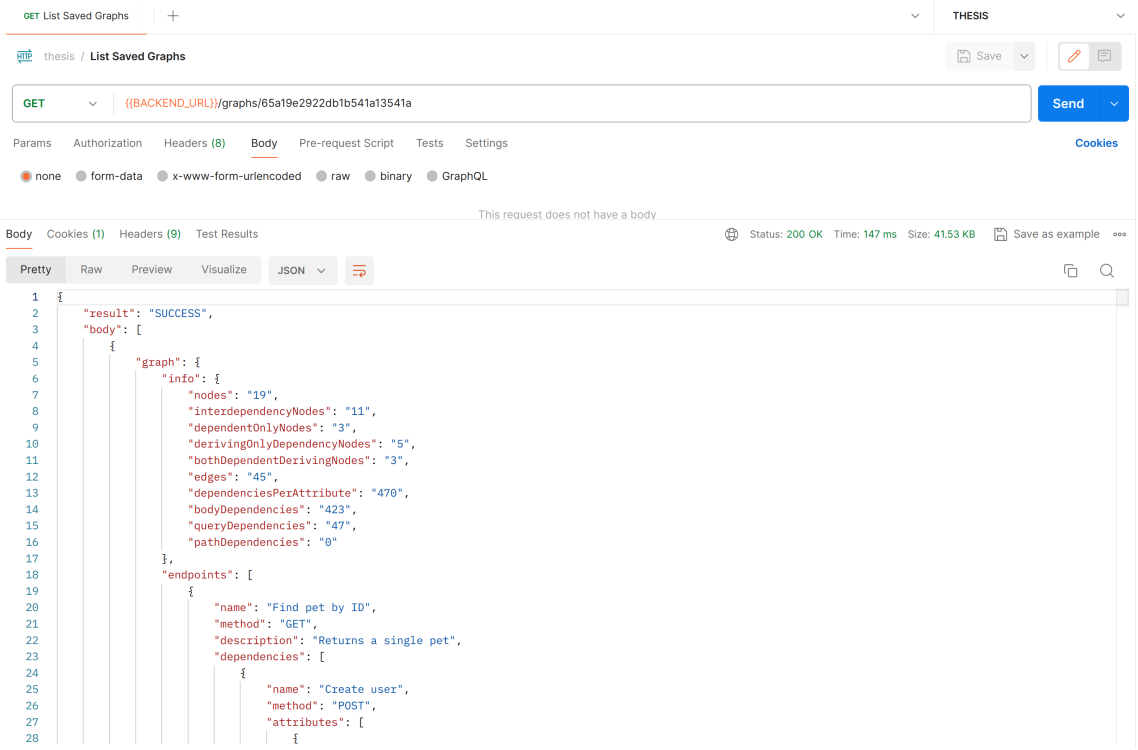
(β') POST /api/token: Το συγκεκριμένο endpoint χρησιμοποιείται για να ελέγχουμε από το frontend app έαν ο χρήστης μας έχει ακόμα active login session. Με το response αυτού του request αποφασίζουμε έαν ο χρήστης μπορεί να δει και να χρησιμοποιήσει functionalities τα οποία είναι διαθέσιμα μόνο σε logged in χρήστες.



Εικόνα 3.22: *Authorize User Request Example*

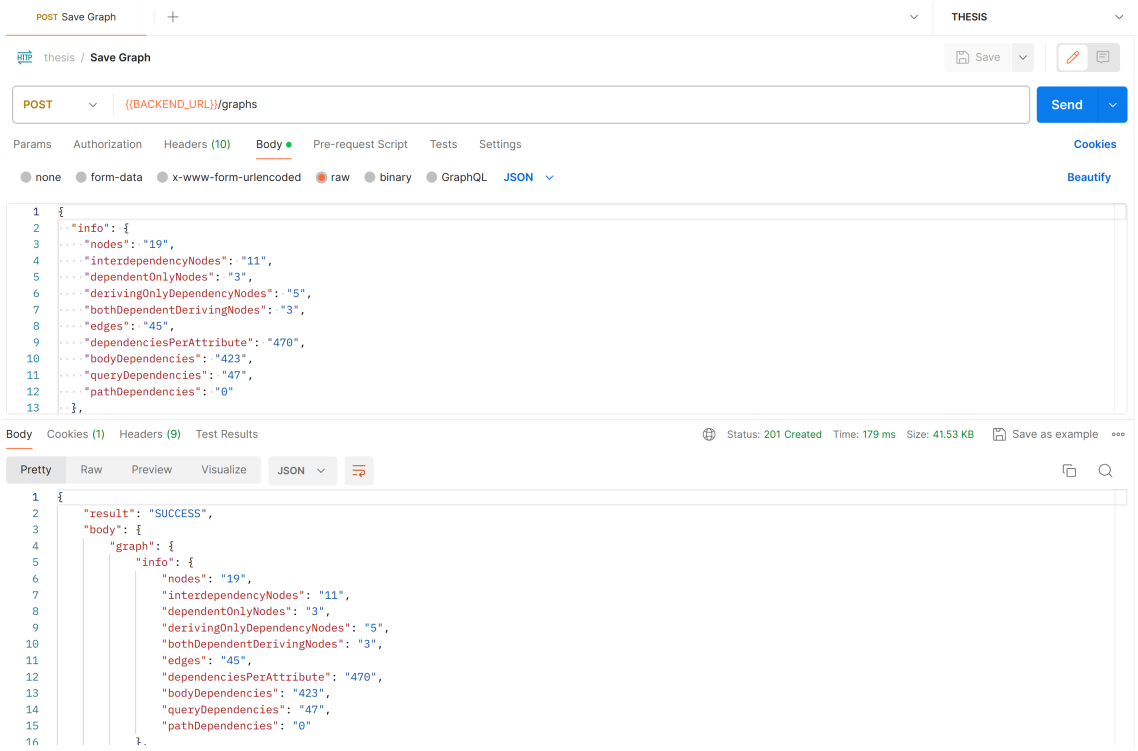
3. Saved Graph Actions: Δημιουργήθηκαν τρία endpoints υπεύθυνα για functionalities που έχουν να κάνουν με τους αποθηκευμένους γράφους. Τα συγκεκριμένα endpoints είναι προστατευμένα με authentication middleware, δηλαδή προορίζονται για εγγεγραμμένους χρήστες με active sessions.

   (α') GET /graphs/:userId: Επιστρέφει όλους τους αποθηκευμένους γράφους ενός χρήστη από τη βάση δεδομένων.



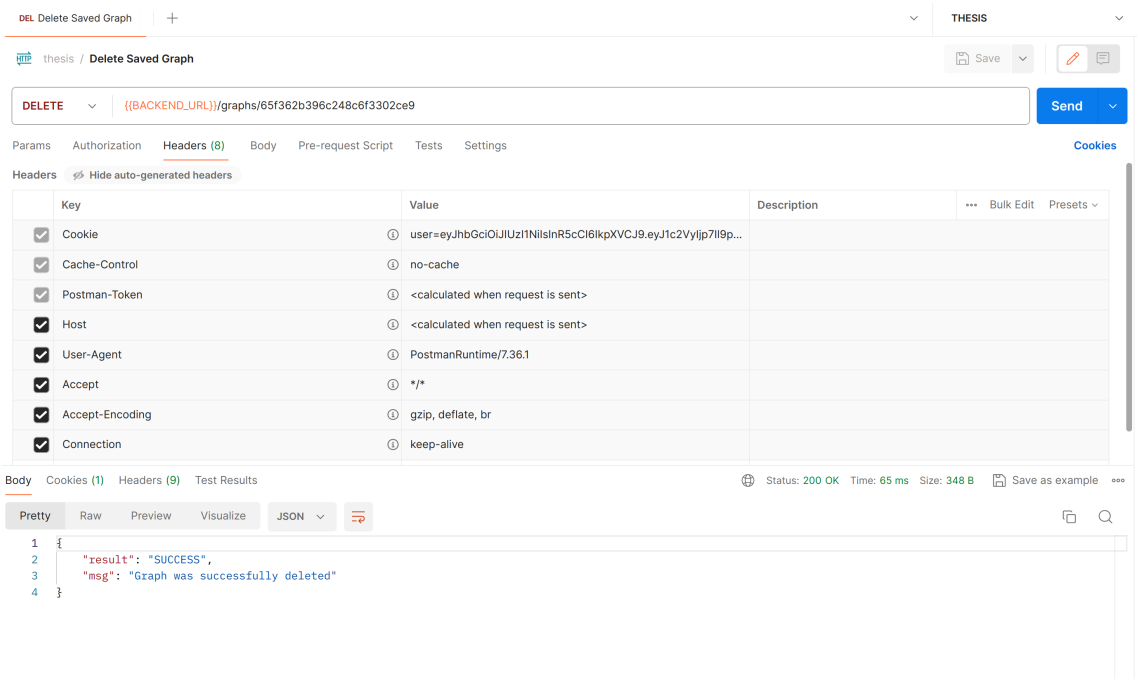
Εικόνα 3.23: *List Saved Graphs Request Example*

   (β') POST /graphs: Αποθηκεύει τα στοιχεία ενός γράφου στη βάση για μελλοντική χρήση.



Εικόνα 3.24: *Save Graph Request Example*

(γ') DELETE /graphs/:graphId: Διαγράφει έναν αποθηκευμένο γράφο ενός χρήστη από τη βάση δεδομένων.



Εικόνα 3.25: *Delete Saved Graph Request Example*

3.4.2 Frontend

Το frontend της εφαρμογής υλοποιήθηκε με τη χρήση των παρακάτω πακέτων και βιβλιοθηκών:

1. react: Η React [28] είναι μία δημοφιλής JavaScript βιβλιοθήκη για την ανάπτυξη διεπαφών χρήστη, η οποία επιτρέπει τη δημιουργία δυναμικών και αποδοτικών Single Page Applications(SPAs). Προσφέρει σημαντικά οφέλη όπως η δημιουργία επαναχρησιμοποιήσιμων UI components, η αποδοτική ενημέρωση του DOM για βελτιωμένη απόδοση, και τη δυνατότητα διαχείρισης της κατάστασης της εφαρμογής. Η ευελιξία και η ευρεία κοινότητα υποστήριξης, μαζί με το πλούσιο οικοσύστημα εργαλείων και βιβλιοθηκών, καθιστούν τη React την πιο δημοφιλή επιλογή για front-end developers που επιδιώκουν να δημιουργήσουν σύγχρονες, διαδραστικές και υψηλής ποιότητας web εφαρμογές.

2. typescript: Η TypeScript [29] είναι μια δυνατή και τυπική επέκταση της JavaScript, που προσθέτει προαιρετικούς τύπους, κλάσεις και διεπαφές, με στόχο την παροχή μιας πιο ασφαλούς και ευκολότερης ανάπτυξης εφαρμογών. Η χρήση της με React είναι δημοφιλής διότι βελτιώνει την ανάπτυξη μέσω της πρόσθεσης στατικών τύπων. Αυτό διευκολύνει την ανίχνευση λαθών κατά την ανάπτυξη, προσφέρει καλύτερη αυτοματοποιημένη ολοκλήρωση κώδικα και βελτιώνει τη συντήρηση και την κλιμακωσιμότητα των μεγάλων εφαρμογών.

3. @reduxjs/toolkit: Το Redux [30] είναι μια βιβλιοθήκη JavaScript για τη διαχείριση και κεντρικοποίηση της κατάστασης εφαρμογών web. Επιτρέπει την ανάπτυξη εφαρμογών που συμπεριφέρονται συνεπώς σε διάφορα περιβάλλοντα (client, server, native), διευκολύνοντας τη δοκιμή και την ανάπτυξη. Χρησιμοποιείται συχνά με React για την αποτελεσματική διαχείριση της κατάστασης UI και την υλοποίηση πιο προβλέψιμων ροών δεδομένων.

4. axios: Το axios [31] είναι μια δημοφιλής βιβλιοθήκη JavaScript που χρησιμοποιείται για την εκτέλεση HTTP αιτήσεων από node.js ή XMLHttpRequests από τον browser, προσφέροντας μια πιο απλή και καθαρή σύνταξη σε σύγκριση με τον προεπιλεγμένο τρόπο που προσφέρει η JavaScript. Είναι promise-based, γεγονός που διευκολύνει τη διαχείριση ασύγχρονου κώδικα και ενσωματώνει αυτόματη μετατροπή JSON δεδομένων. Μέσω του axios επιτυγχάνεται η επικοινωνία του frontend application με το backend.

5. @react-oauth/google: Το @react-oauth/google [32] είναι μια βιβλιοθήκη για React που διευκολύνει την ενσωμάτωση της διαδικασίας ελέγχου ταυτότητας (authentication) με την χρήση του Google OAuth. Αυτή η βιβλιοθήκη παρέχει components και hooks που κάνουν πιο απλή την υλοποίηση της σύνδεσης μέσω Google σε εφαρμογές React, διαχειριζόμενη την ασφάλεια και την ανταλλαγή δεδομένων με το Google OAuth API. Με τη χρήση αυτού του πακέτου επιτυγχάνουμε το authentication των χρηστών της εφαρμογής μας.

6. reagraph: Το reagraph [33] είναι ένα high-performance graph visualization tool χτισμένο σε WebGL. Αποτελεί τον πυρήνα της εφαρμογής μας. Με τη χρήση του πακέτου επιτυγχάνεται η οπτικοποίηση των εξαρτήσεων μεταξύ των endpoints. Είναι highly customizable, προσφέρει πολλές λειτουργίες out-of-the box και μπορεί να οπτικοποιήσει μεγάλο όγκο δεδομένων. Πολύ σημαντικό επίσης αποτελεί και το γεγονός ότι είναι open-source πακέτο. Συγκεκριμένα, χρειαζόμασταν ορισμένες λειτουργίες τις οποίες δε τις υποστήριζε ακόμα το πακέτο αλλά καταφέραμε να τις υλοποιήσουμε κάνοντας development στο πακέτο και ενσωματώνοντάς τες στην εφαρμογή μας.

7. patch-package: Το patch-package [34] είναι ένα εργαλείο που επιτρέπει στους developers να κάνουν διορθώσεις ή προσθήκες (patches) σε ήδη εγκατεστημένα npm packages μέσα στο project τους. Αυτό είναι ιδιαίτερα χρήσιμο όταν ένας developer εντοπίζει ένα bug σε ένα dependency της εφαρμογής του και χρειάζεται να εφαρμόσει μια γρήγορη διόρθωση πριν η επίσημη διόρθωση γίνει διαθέσιμη από τους δημιουργούς του package. Το patch-package διευκολύνει τη διαδικασία δημιουργίας και εφαρμογής αυτών των διορθώσεων και προσθηκών, διασφαλίζοντας ότι οι αλλαγές παραμένουν μέρος του project ακόμα και μετά από επανεγκαταστάσεις των npm packages. Μέσω αυτού του πακέτου, ενσωματώνουμε τις λειτουργίες που χρειαζόμαστε αλλά δεν υποστηρίζει το reagraph πακέτο στην εφαρμογή μας.

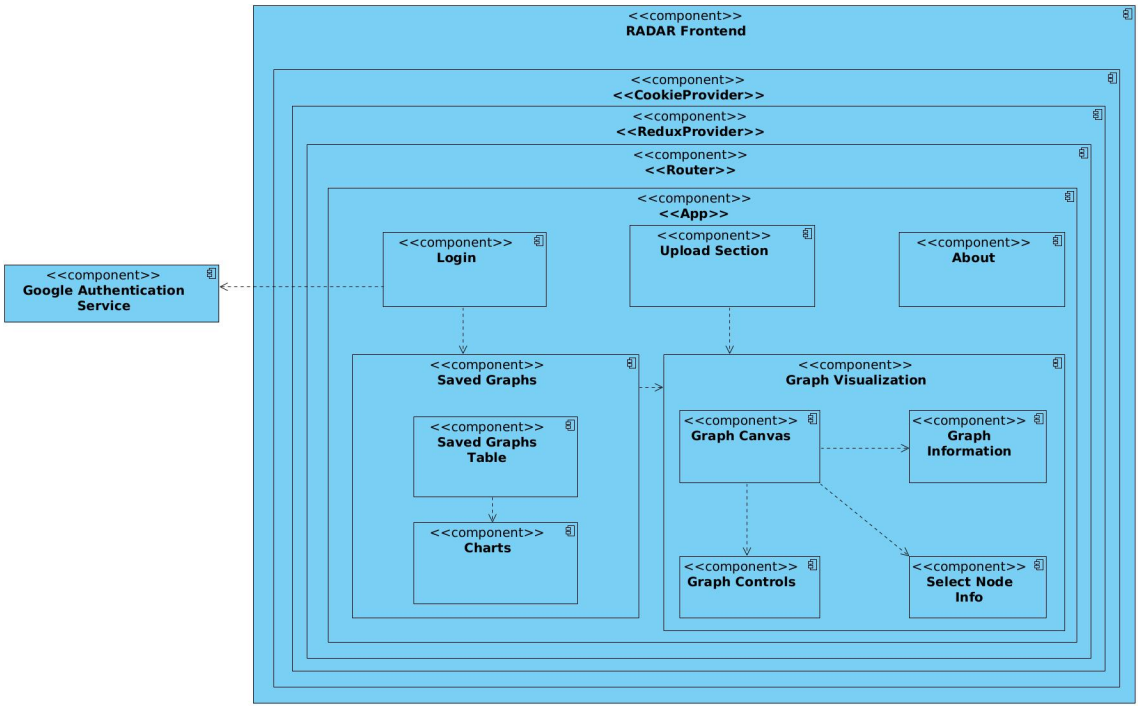
8. ybug-react: Το ybug-react [35] είναι ένα εργαλείο το οποίο δίνει τη δυνατότητα στους χρήστες της εφαρμογής να αποτυπώσουν κάποιο visual bug ή να αφήσουν ένα comment σχετικά με κάτι που μπορεί να χρειάζεται βελτίωση με εύκολο και γρήγορο τρόπο και να μας ενημερώσουν σχετικά.

9. highcharts-react-official: Το highcharts-react-official [36] είναι η επίσημη βιβλιοθήκη React για το Highcharts, που είναι μια βιβλιοθήκη JavaScript για τη δημιουργία διαγραμμάτων. Διευκολύνει την ενσωμάτωση διαφόρων τύπων διαγραμμάτων σε εφαρμογές React, επιτρέποντας την αποδοτική και εύκολη δημιουργία πλούσιων οπτικοποιήσεων δεδομένων. Με τη χρήση του πακέτου, παράγουμε γραφήματα με τα δεδομένα που προκύπτουν από την ανάλυση των εξαρτήσεων μεταξύ των endpoints.

10. react-toastify: Το react-toastify [37] είναι μια βιβλιοθήκη React που επιτρέπει την εύκολη δημιουργία και διαχείριση ειδοποιήσεων (toasts) μέσα σε μια εφαρμογή. Παρέχει μια απλή διεπαφή για την προσθήκη ειδοποιήσεων με πλούσιες δυνατότητες προσαρμογής, όπως τη διαμόρφωση της διάρκειας εμφάνισης, τη θέση, και την εμφάνιση των toasts. Τη χρησιμοποιούμε για να ενημερώσουμε τους χρήστες της εφαρμογής για τα αποτελέσματα διαφόρων ενεργειών, όπως η επιτυχής αποθήκευση ενός γράφου ή ένος σφάλματος που προέκυψε κατά την διαδικασία upload κάποιου αρχείου.

11. TailwindCss: Το TailwindCSS [38] είναι ένα utility-first CSS framework για την κατασκευή custom διεπαφών χρήστη γρήγορα και με αποδοτικότητα, χωρίς να χρειάζεται custom CSS. Προσφέρει μια ευρεία γκάμα από utility classes για σχεδόν κάθε CSS property, επιτρέποντας την δημιουργία responsive layouts με ταχύτητα, ευελιξία και με έμφαση στην επαναχρησιμοποίηση κώδικα.

12. DaisyUI: Το DaisyUI [39] είναι ένα plugin για το TailwindCSS που προσφέρει ένα σύνολο από προσχεδιασμένα components, όπως κουμπιά, φόρμες και κάρτες, διευκολύνοντας την ανάπτυξη όμορφων και συνεπών διεπαφών χρήστη. Επιτρέπει στους developers να δημιουργούν ελκυστικά UI με μικρότερο κόπο, χρησιμοποιώντας τις utility classes του TailwindCSS για την προσαρμογή των στοιχείων. Η εφαρμογή μας είναι υλοποιημένη με χρήση TailwindCss και χρησιμοποιεί πολλά components από το DaisyUI plugin.

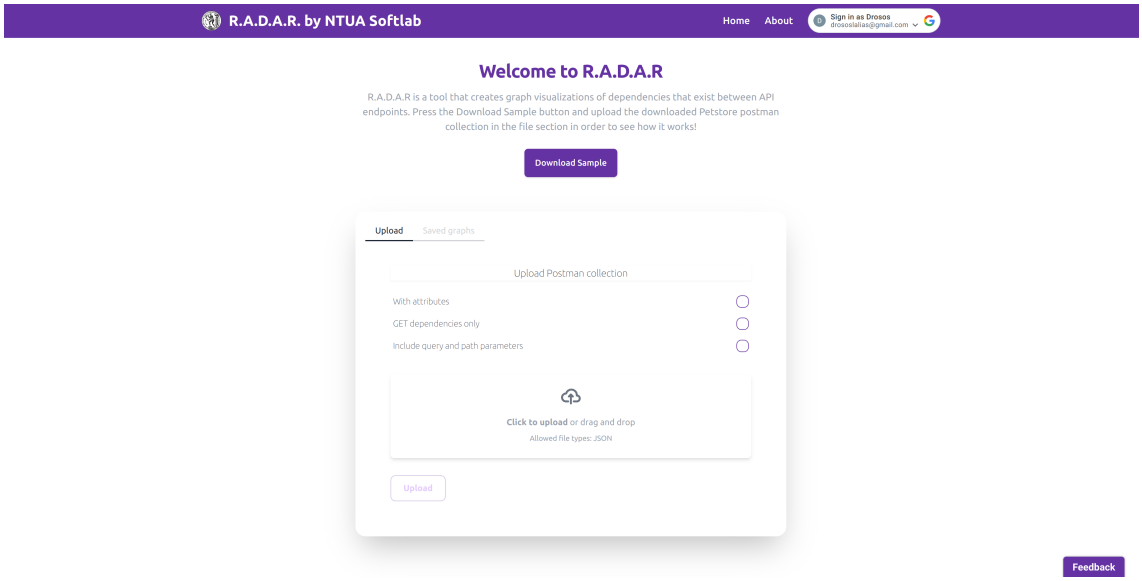


Εικόνα 3.26: *Frontend Component Diagram of RADAR*

3.5 Επίδειξη frontend

3.5.1 Upload File Form

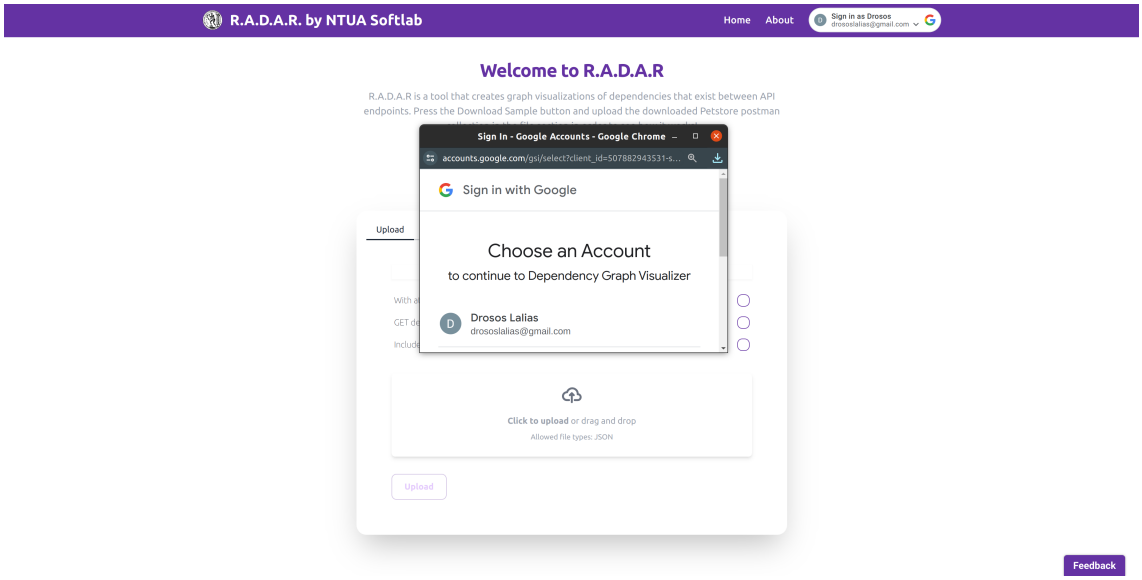
Η αρχική σελίδα του RADAR όπου ο χρήστης μπορεί να ανεβάσει Postman Collection αρχεία.



Εικόνα 3.27: *Upload File Form*

3.5.2 Login Modal

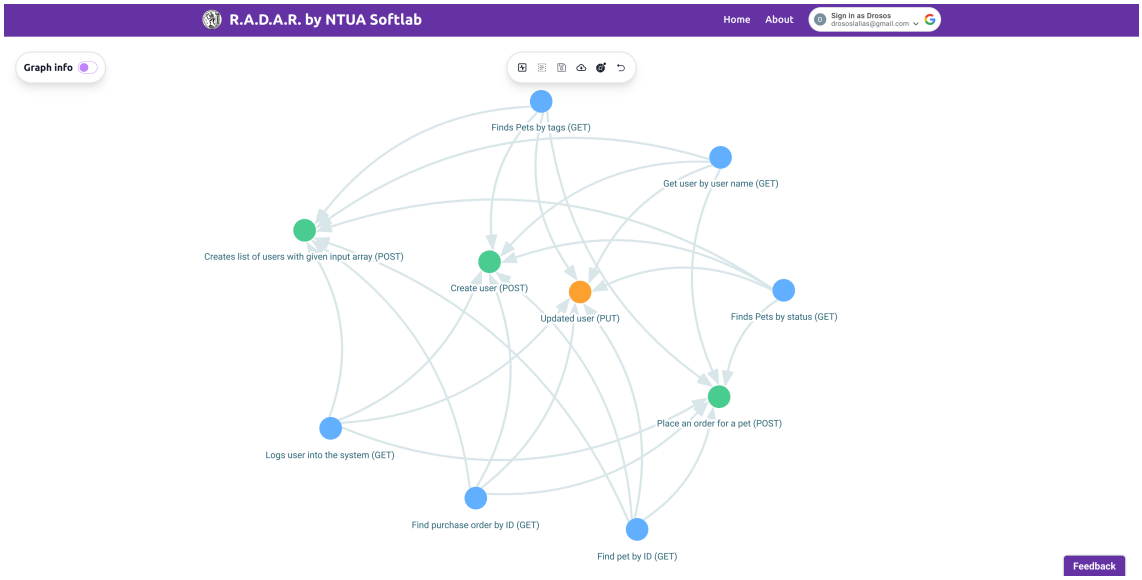
Login Modal όπου ο χρήστης επιλέγει το λογαριασμό Google με τον οποίο θέλει να συνδεθεί στην εφαρφογή.



Εικόνα 3.28: *Login Modal*

3.5.3 Graph Visualization

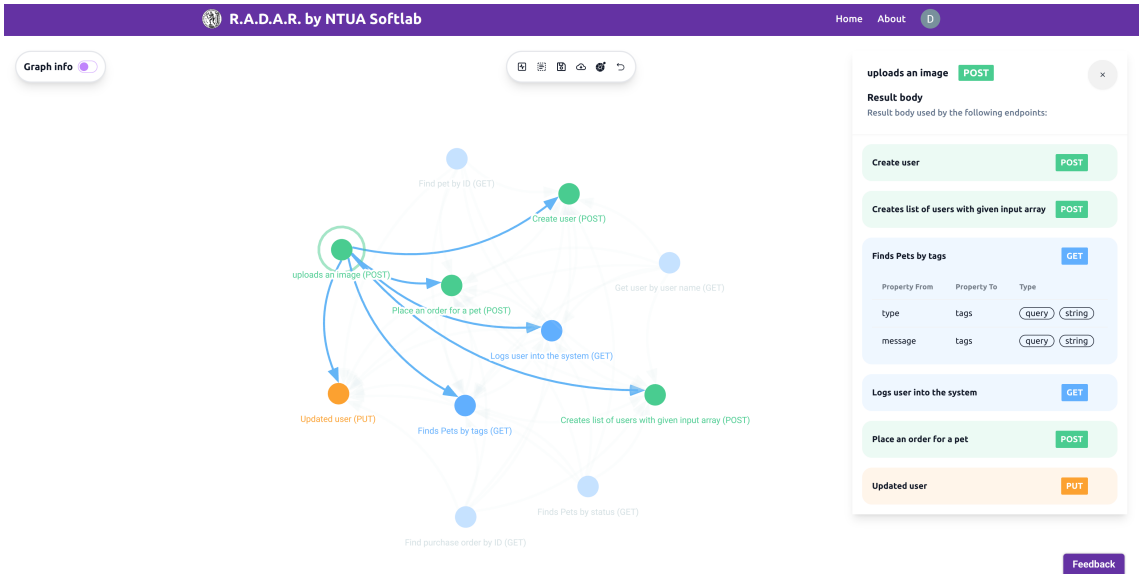
Η πιο σημαντική σελίδα του RADAR η οποία εμφανίζει τον γράφο εξαρτήσεων.



Εικόνα 3.29: *Graph Visualization*

3.5.4 Graph Visualization Selected Node

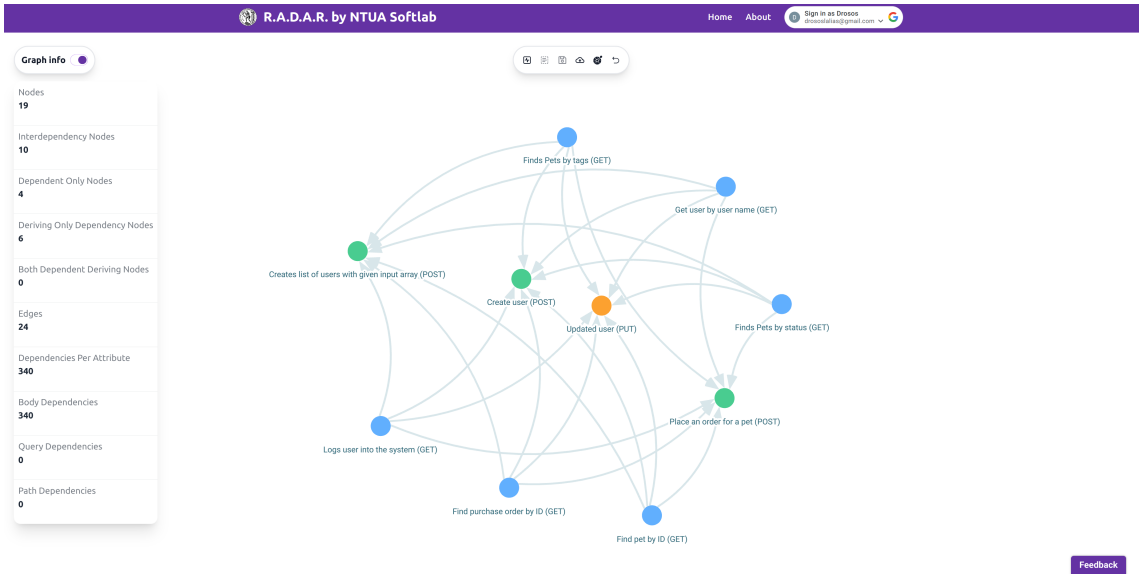
Μόλις ο χρήστης επιλέξει κάποιο node, εμφανίζεται στα δεξιά ένα section το οποίο είναι της μορφής swagger και περιέχει τις εξαρτήσεις του επιλεγμένου node.



Εικόνα 3.30: *Graph Visualization Selected Node*

3.5.5 Graph Visualization Graph Info

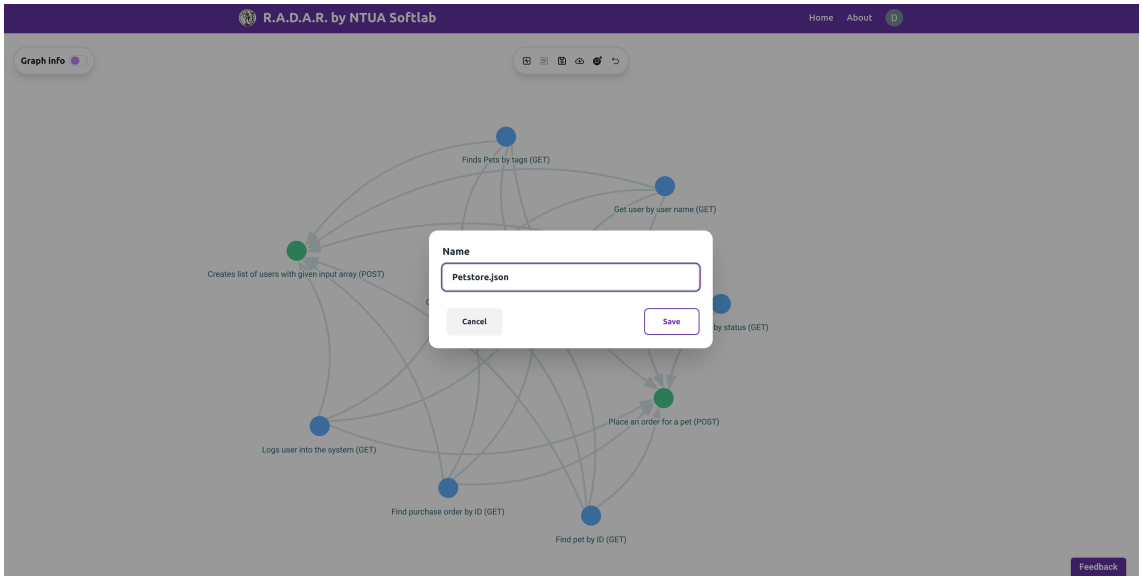
Στην παρακάτω σελίδα φαίνονται ορισμένες πληροφορίες οι οποίες προκύπτουν κατά την ανάλυση εξαρτήσεων του input file.



Εικόνα 3.31: *Graph Visualization Graph Info*

3.5.6 Graph Visualization Save Modal

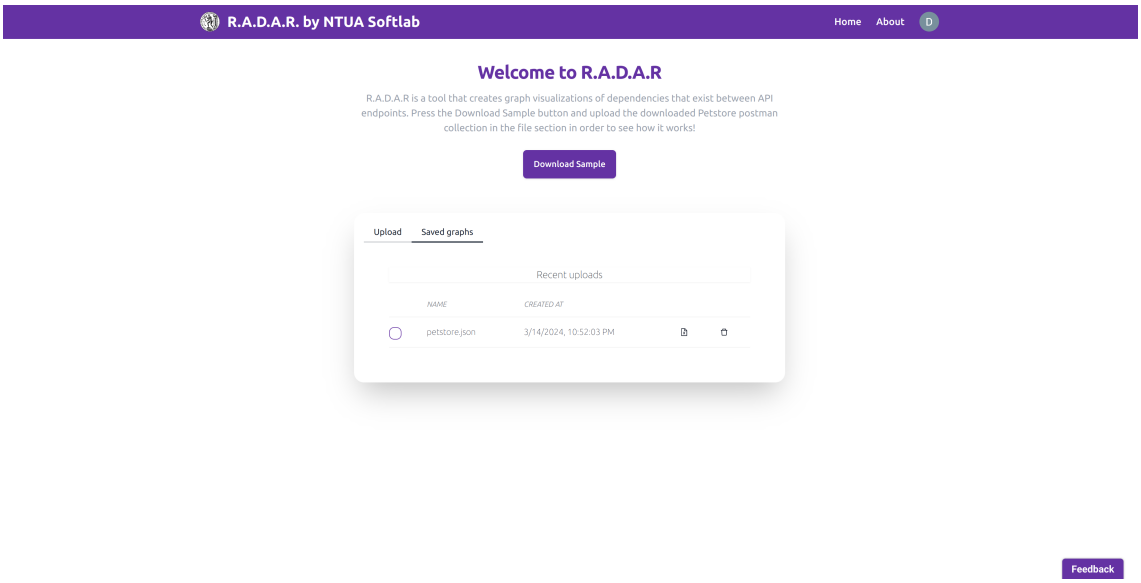
Το παρακάτω modal εμφανίζεται μόλις ο χρήστης επιλέξει να κάνει Save τον γράφο για μελλοντική χρήση.



Εικόνα 3.32: *Graph Visualization Save Modal*

3.5.7 Saved Graphs Tab

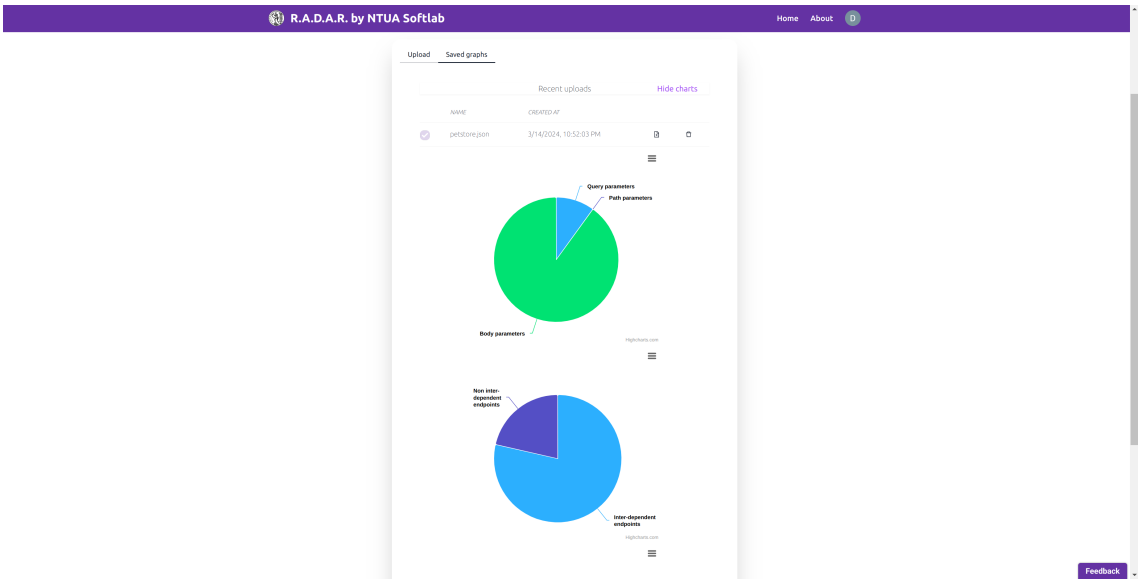
Στην παρακάτω σελίδα εμφανίζεται ο πίνακας με όλους τους αποθηκευμένους γράφους του χρήστη.



Εικόνα 3.33: *Saved Graphs Tab*

3.5.8 Saved Graphs Charts

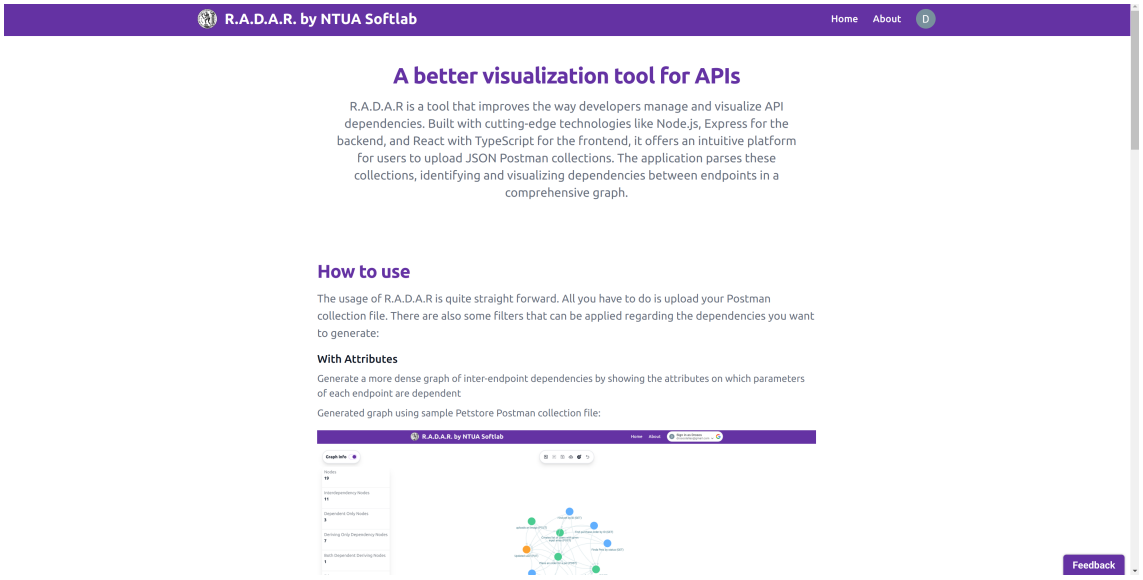
Στην παρακάτω σελίδα παρουσιάζονται σε διάγραμμα οι πληροφορίες που προέκυψαν κατά την ανάλυση των εξαρτήσεων.



Εικόνα 3.34: *Saved Graphs Charts*

3.5.9 About Page

Η παρακάτω σελίδα περιγράφει τον τρόπο λειτουργίας του RADAR.



Εικόνα 3.35: *About Page*

3.6 Περιπτώσεις Εκτέλεσης

Στην τρέχουσα ενότητα θα παρουσιάσουμε κάποιες περιπτώσεις εκτέλεσης της εφαρμογής που υλοποιήσαμε χρησιμοποιώντας τα Postman collections από ορισμένα δημοφιλή public APIs. Τα APIs τα οποία θα χρησιμοποιήσουμε είναι:

- BoxPlatform API: Το Box Platform API είναι ένα cloud-based API που παρέχεται από την εταιρεία Box, η οποία είναι γνωστή για την παροχή υπηρεσιών αποθήκευσης και διαχείρισης αρχείων στο cloud. Το Box Platform API σχεδιάστηκε για να επιτρέπει στους προγραμματιστές να ενσωματώσουν τις δυνατότητες αποθήκευσης, διαχείρισης και κοινής χρήσης αρχείων του Box στις δικές τους εφαρμογές ή πλατφόρμες. Με αυτό τον τρόπο, το Box Platform API επεκτείνει την υπηρεσία αποθήκευσης αρχείων σε μια πλατφόρμα ανάπτυξης λογισμικού (Software Development Platform), προσφέροντας μια σειρά από δυνατότητες για την ασφαλή διαχείριση και πρόσβαση στα αρχεία και τα δεδομένα.

- Notion API: Το Notion API είναι ένα API που παρέχεται από το Notion, μια εφαρμογή οργάνωσης και συνεργασίας που επιτρέπει στους χρήστες να δημιουργούν, να μοιράζονται και να διαχειρίζονται σημειώσεις, βάσεις δεδομένων, ημερολόγια και άλλα είδη περιεχομένου. Μέσω του Notion API, οι προγραμματιστές και οι επιχειρήσεις μπορούν να αυτοματοποιήσουν εργασίες, να ενσωματώσουν το Notion με άλλες εφαρμογές και συστήματα, και να δημιουργήσουν προσαρμοσμένες λύσεις που εκμεταλλεύονται τα δεδομένα και τις δυνατότητες του Notion.

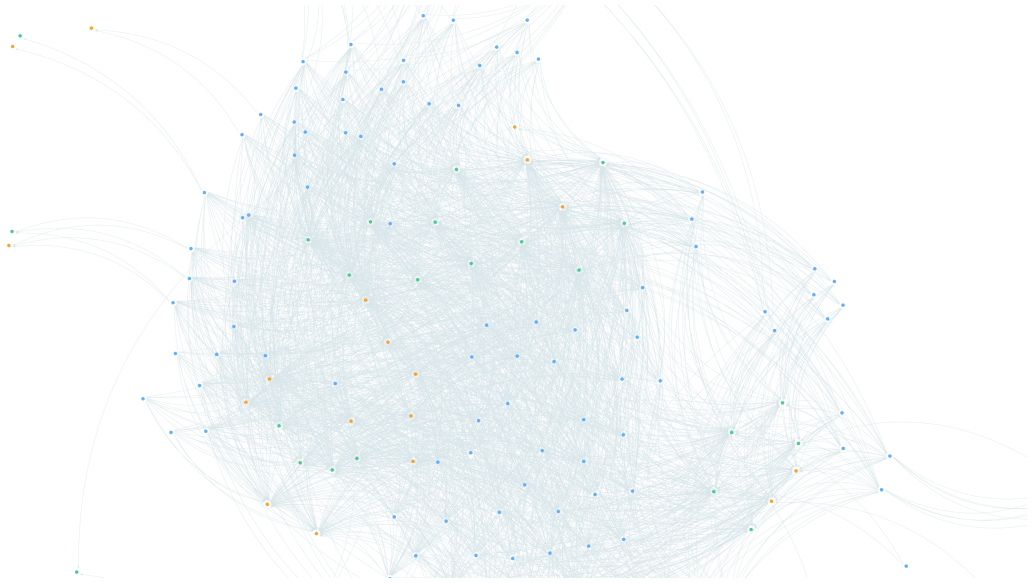
- OpenAI API: Το OpenAI API είναι ένα API που παρέχεται από την OpenAI, μια εταιρεία που εστιάζει στην έρευνα και την ανάπτυξη στον τομέα της τεχνητής νοημοσύνης. Το OpenAI API δίνει πρόσβαση σε μια σειρά από προηγμένα μοντέλα τεχνητής νοημοσύνης, συμπεριλαμβανομένων των γεννήτριων φυσικής γλώσσας, όπως το GPT (Generative Pre-trained Transformer).

- Paypal API: Το PayPal API είναι ένα API που παρέχεται από την PayPal, μια από τις πιο διαδεδομένες υπηρεσίες ηλεκτρονικών πληρωμών παγκοσμίως. Το API επιτρέπει στους προγραμματιστές να ενσωματώσουν τη δυνατότητα επεξεργασίας πληρωμών μέσω PayPal σε ιστοσελίδες, εφαρμογές κινητών και άλλες ψηφιακές πλατφόρμες. Μέσω του PayPal API, είναι δυνατή η αυτοματοποίηση εργασιών όπως η δημιουργία και η διαχείριση πληρωμών, η ανάληψη κεφαλαίων, η διαχείριση λογαριασμών και η ανταλλαγή χρηματοοικονομικών δεδομένων.

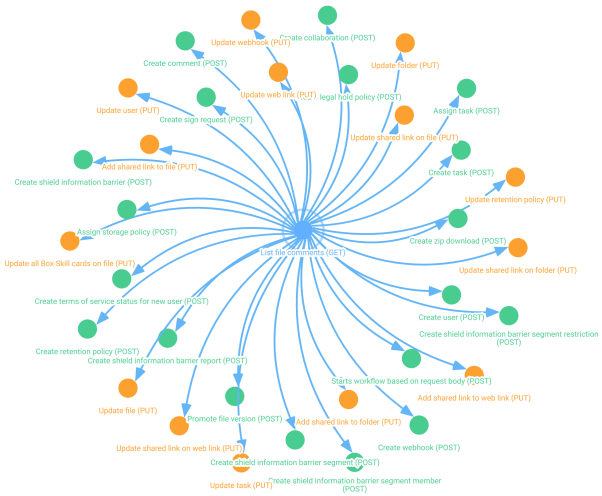
- ZoomMeeting API: Το ZoomMeeting API αποτελεί μέρος του Zoom API, το οποίο παρέχεται από την εταιρεία Zoom Video Communications για τη διαχείριση και την αυτοματοποίηση των λειτουργιών συνεδριάσεων στην πλατφόρμα της. Το Zoom API επιτρέπει στους προγραμματιστές να ενσωματώνουν τις δυνατότητες του Zoom σε εφαρμογές, ιστοσελίδες και άλλα συστήματα, επιτρέποντας τη διαχείριση συναντήσεων, την προγραμματισμένη δημιουργία συναντήσεων, την πρόσκληση συμμετεχόντων και άλλες σχετικές δραστηριότητες μέσω του προγραμματισμού.

Η μορφή του documentation των παραπάνω APIs είναι διαθέσιμα σε μορφή ιστοσελίδας (web-based documentation), παρέχοντας μια εύχρηστη και πλούσια σε πληροφορίες πηγή για προγραμματιστές και χρήστες που επιθυμούν να εξερευνήσουν ή να τα χρησιμοποιήσουν. Η ιστοσελίδα της τεκμηρίωσής τους περιλαμβάνει οδηγούς εκκίνησης (getting started guides), πληροφορίες για τις διαθέσιμες κλήσεις API, παραδείγματα κώδικα, αναφορές API για τα διαφορετικά endpoints, καθώς και οδηγίες για την ασφάλεια, τη διαχείριση των δικαιωμάτων πρόσβασης και την καλύτερη χρήση τους.

3.6.1 Περίπτωση Εκτέλεσης 1: BoxPlatform

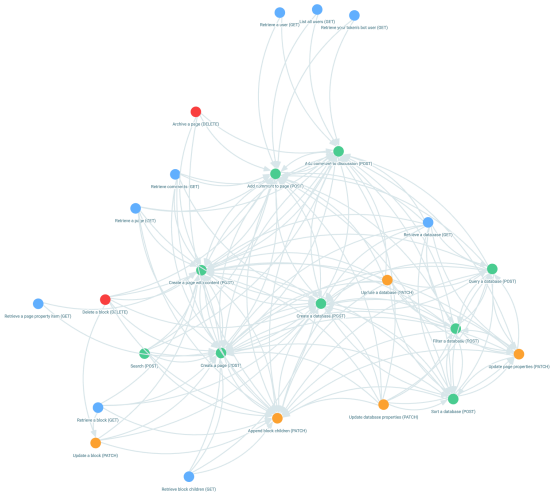


Εικόνα 3.36: *BoxPlatform Visualization*



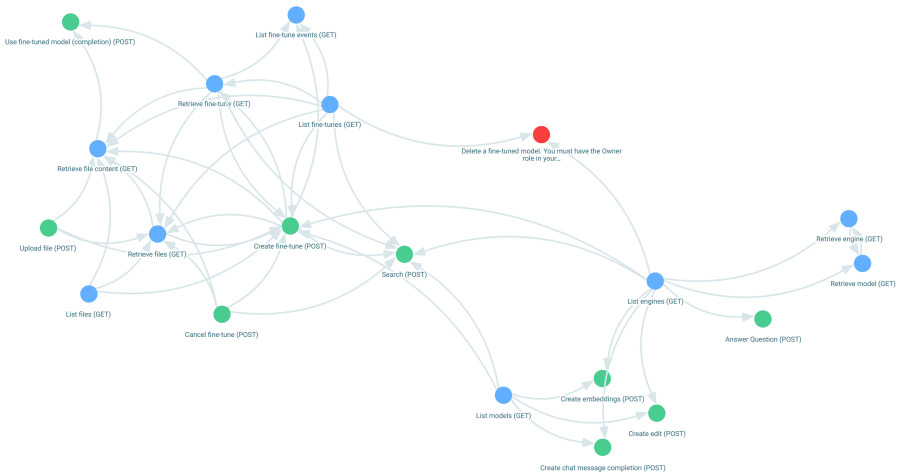
Εικόνα 3.37: *BoxPlatform Visualization - Isolated Node*

3.6.2 Περίπτωση Εκτέλεσης 2: NotionAPI



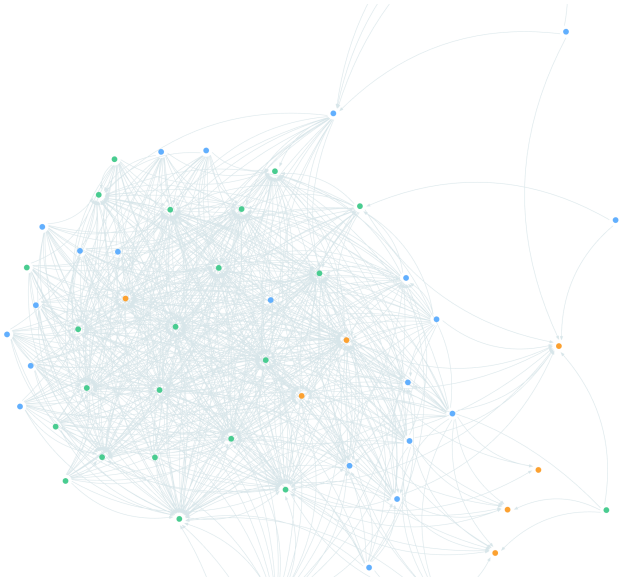
Εικόνα 3.38: *NotionAPI Visualization*

3.6.3 Περίπτωση Εκτέλεσης 3: OpenAI

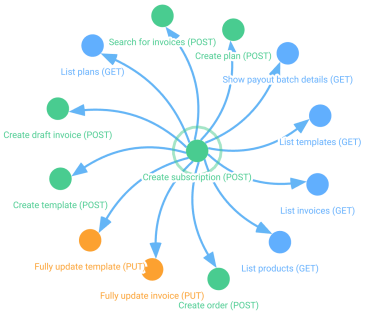


Εικόνα 3.39: *OpenAI Visualization*

3.6.4 Περίπτωση Εκτέλεσης 4: Paypal

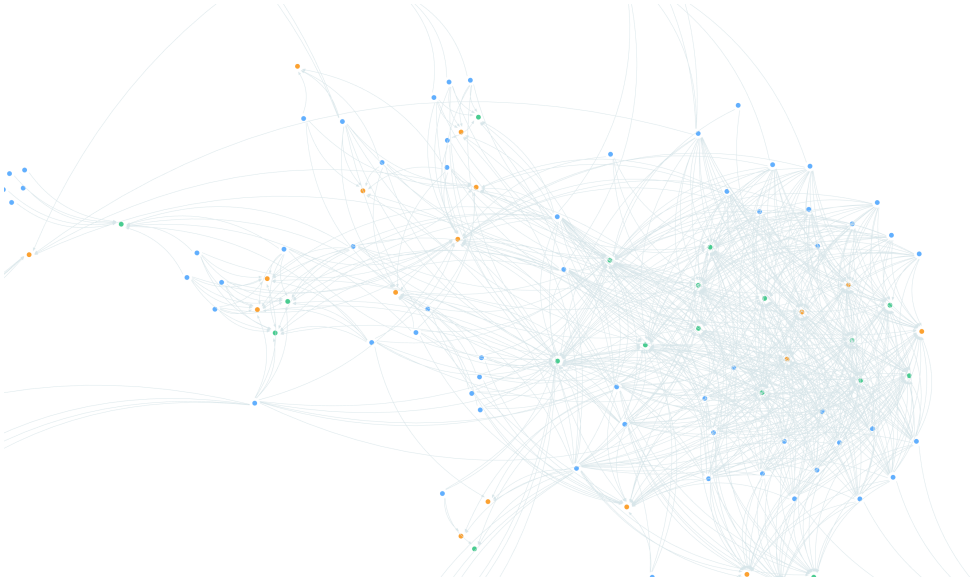


Εικόνα 3.40: *Paypal Visualization*

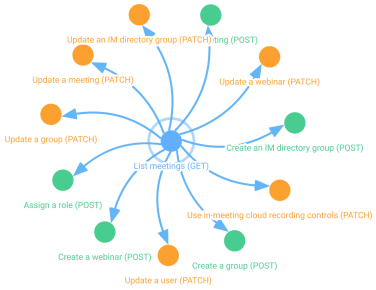


Εικόνα 3.41: *Paypal Visualization - Isolated Node*

3.6.5 Περίπτωση Εκτέλεσης 5: ZoomMeeting



Εικόνα 3.42: *ZoomMeeting Visualization*



Εικόνα 3.43: *ZoomMeeting Visualization - Isolated Node*

Κεφάλαιο 4

Επίλογος

4.1 Μελέτη Αποτελεσμάτων

Η εφαρμογή που αναπτύξαμε, όπως είδαμε στις περιπτώσεις εκτέλεσης, προσφέρει δυνατότητες οπτικοποίησης για την ανάλυση και την παρουσίαση των εξαρτήσεων μεταξύ των endpoints ενός API. Ωστόσο, αντιμετωπίζει ορισμένους περιορισμούς, ιδίως όταν το μέγεθος και η πολυπλοκότητα του API αυξάνονται, όπως στην περίπτωση του Box Platform, το οποίο αποτελείται από περίπου 250 nodes και 15000 edges. Αυτοί οι περιορισμοί μπορεί να επηρεάσουν την αλληλεπίδραση του χρήστη με τον γράφο, απαιτώντας περαιτέρω βελτιώσεις για την υποστήριξη τέτοιων μεγάλων και σύνθετων API.

Παρά τους περιορισμούς για APIs με τεράστιο αριθμό endpoints, το R.A.D.A.R μπορεί να υποστηρίξει μεγάλο εύρος APIs και να συμβάλει σημαντικά σε διαφόρους τομείς, όπως:

1. API Development και Testing: Οι προγραμματιστές ανεβάζουν τις συλλογές Postman για να οπτικοποιήσουν τις εξαρτήσεις μεταξύ των endpoints των API τους, βοηθώντας στην ταυτοποίηση περιττών κλήσεων ή στην βελτιστοποίηση των ακολουθιών κλήσεων API για αποδοτικότητα.

2. Documentation και Onboarding: Νέα μέλη της ομάδας ή εξωτερικοί συνεργάτες χρησιμοποιούν την εφαρμογή για να καταλάβουν γρήγορα τη δομή και τη ροή δεδομένων του API, βελτιώνοντας τη συνεργασία και μειώνοντας τον χρόνο εκπαίδευσης.

3. Integration Planning: Αρχιτέκτονες συστημάτων και διαχειριστές έργων ανεβάζουν συλλογές για να σχεδιάσουν και να επαληθεύσουν την ενσωμάτωση νέων υπηρεσιών ή αλλαγών στο σύστημα, εξασφαλίζοντας τη συμβατότητα και μειώνοντας τα ζητήματα ενσωμάτωσης.

4. Troubleshooting και Debugging: Κατά τη διάρκεια ζητημάτων με τη συμπεριφορά του API, οι χρήστες μπορούν να αναλύσουν τις οπτικοποιημένες εξαρτήσεις για να εντοπίσουν προβληματικές αλληλεπιδράσεις ή ανεπιθύμητες αλυσίδες εξαρτήσεων.

5. Εκπαιδευτικοί Σκοποί: Εκπαιδευτές που διδάσκουν web development ή αρχιτεκτονική συστημάτων μπορούν να χρησιμοποιήσουν την εφαρμογή για να δείξουν στους φοιτητές πραγματικές αλληλεπιδράσεις και εξαρτήσεις API, ενισχύοντας τη μάθηση μέσω οπτικών βοηθημάτων.

Αυτές οι περιπτώσεις χρήσης της εφαρμογής τονίζουν την αξία της εφαρμογής στη βελτίωση των πρακτικών ανάπτυξης API, την ενίσχυση της ομαδικής συνεργασίας και τη διευκόλυνση της μάθησης.

4.2 Συμπεράσματα

Η ολοκλήρωση αυτής της διπλωματικής εργασίας έχει αποκαλύψει με εμφατικό τρόπο την ανεκτίμητη αξία της ανάλυσης και της οπτικοποίησης των εξαρτήσεων μεταξύ των endpoints ενός API στην ανάπτυξη λογισμικού. Η δυνατότητα απεικόνισης αυτών των πολυπλοκοτήτων σε ένα γράφο δεν απλοποιεί μόνο την κατανόηση των διασυνδέσεων, αλλά παρέχει και σημαντικές πληροφορίες για την αποτελεσματικότητα, την ασφάλεια και τη βελτιστοποίηση των εφαρμογών. Μέσα από την πρακτική αυτή εφαρμογή, η εργασία υπογραμμίζει την ανάγκη για συνεχή έρευνα και καινοτομία στην τεχνολογία λογισμικού, αναδεικνύοντας παράλληλα τη σημασία της ενσωμάτωσης νέων εργαλείων και τεχνικών στη διαδικασία ανάπτυξης. Η συμβολή της στον τομέα δεν περιορίζεται μόνο στην ανάπτυξη ενός εργαλείου ανάλυσης, αλλά επεκτείνεται και στην ενθάρρυνση μιας πιο διερευνητικής και κριτικής σκέψης στην κοινότητα των developers, καθιστώντας την ένα σημαντικό βήμα προς την πρόοδο και την καινοτομία στον κόσμο της ανάπτυξης λογισμικού.

4.3 Μελλοντικές Επεκτάσεις

Το σύστημα που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας θα μπορούσε να βελτιωθεί και να επεκταθεί περαιτέρω, τουλάχιστον ως προς τρεις κατευθύνσεις. Συγκεκριμένα, αναφέρονται τα ακόλουθα:

- Δημιουργία plugin για το Postman. Το integration του R.A.D.A.R με το Postman ως plugin θα επέτρεπε την αυτόματη ανάλυση και οπτικοποίηση των εξαρτήσεων απευθείας από το εργαλείο που χρησιμοποιούν ήδη οι developers για τη διαχείριση των API τους, παρέχοντας μια άμεση και ενσωματωμένη εμπειρία.

- Υποστήριξη περισσότερων τύπων αρχείων. Η υποστήριξη περισσότερων τύπων αρχείων, όπως YAML, θα προσέδιδε σημαντική ευελιξία και θα διεύρυνε την προσβασιμότητα του εργαλείου σε διαφορετικά περιβάλλοντα ανάπτυξης. Αυτό επιτρέπει στους χρήστες να αναλύσουν και να οπτικοποιήσουν εξαρτήσεις API από μια ευρύτερη γκάμα πηγών και format, καθιστώντας το εργαλείο πιο πολυσύνθετο και προσαρμοστικό στις ανάγκες των σύγχρονων προγραμματιστών και μηχανικών λογισμικού.

- Δυνατότητα sharing αποθηκευμένων γράφων. Η δυνατότητα sharing του αποθηκευμένου γράφου σε άλλον χρήστη της εφαρμογής θα ενίσχυε τη συνεργασία και την κοινή χρήση γνώσεων μεταξύ των developers, διευκολύνοντας την ανταλλαγή πολύτιμων πληροφοριών για την ανάλυση και τη βελτίωση των API.

Βιβλιογραφία

[1] Π. Παπαδέας. *Αυτόματη Παραγωγή Τεκμηρίωσης Εξαρτήσεων Μεταξύ Κλήσεων Endpoints ενός REST API*. Πτυχιακή εργασία, Εθνικό Μετσόβιο Πολυτεχνείο, 2023.

[2] Ν. Αμπατζή. *Αυτοματοποίηση Λειτουργίας και Documentation API*. Πτυχιακή εργασία, Εθνικό Μετσόβιο Πολυτεχνείο, 2022.

[3] *History of APIs*. https://en.wikipedia.org/wiki/API.

[4] *What is an API*. https://bigblue.academy/gr/ti-einai-to-api.

[5] *REST API Principles*. https://aws.amazon.com/what-is/restful-api/.

[6] *REST API Essential Methods*. https://www.techtarget.com/searchapparchitecture/tip/The-5-essential-HTTP-methods-in-RESTful-API-development.

[7] *What are api parameters*. https://apipheny.io/what-are-api-parameters/.

[8] *SOAP API*. https://www.indeed.com/career-advice/career-development/what-is-soap-api.

[9] *SOAP vs REST*. https://www.interviewbit.com/blog/soap-vs-rest/.

[10] *SOAP vs REST APIs*. https://blog.postman.com/soap-vs-rest/.

[11] *Swaggerhub*. https://swagger.io/tools/swaggerhub/.

[12] *Stoplight*. https://stoplight.io/.

[13] *Redocly*. https://redocly.com/docs/.

[14] *PlantUML*. https://plantuml.com/.

[15] *UML*. https://en.wikipedia.org/wiki/Unified_Modeling_Language.

[16] *Visual Paradigm*. https://en.wikipedia.org/wiki/Unified_Modeling_Language.

[17] *Class Diagram*. https://en.wikipedia.org/wiki/Class_diagram.

[18] *Use Case Diagram*. https://en.wikipedia.org/wiki/Use_case_diagram.

[19] Hao Zhong, Na Meng, Zexuan Li και Li Jia. *An empirical study on API parameter rules*. *42nd International Conference on Software Engineering (ICSE '20)*, New York, USA, 2020.

[20] Antonia Bertolino, Paola Inverardi, Patrizio Pelliccione και Massimo Tivoli. *An empirical study on API parameter rules*. *In Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (ESEC/FSE '09)*, New York, USA, 2009.

[21] Oostvogels N., De Koster J. και De Meuter W. *Inter-parameter Constraints in Contemporary Web APIs*. *ICWE '17*, New York, USA, 2017.

[22] Alberto Martin-Lopez. *Automated analysis of inter-parameter dependencies in web APIs*. *In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceed- ings (ICSE '20)*, New York, USA, 2020.

[23] Xiaoying Bai, Wenli Dong, Wei Tek Tsai και Yinong Chen. *WSDL-based automatic test case generation for Web services testing*. *IEEE International Workshop on Service-Oriented System Engineering (SOSE'05)*, Beijing, China, 2005.

[24] A. Chaturvedi και D. Binkley. *Web Service Slicing: Intra and Inter- Operational Analysis to Test Changes*. *IEEE Transactions on Services Computing*, 2021.

[25] *TailwindCSS*. https://nodejs.org/en.

[26] *ExpressJS*. https://expressjs.com/.

[27] *MongoDB*. https://www.mongodb.com/.

[28] *ReactJS*. https://react.dev/.

[29] *Typescript*. https://www.typescriptlang.org/.

[30] *Redux*. https://redux.js.org/.

[31] *Axios*. https://https://github.com/axios/axios.

[32] *React OAuth2 | Google*. https://www.npmjs.com/package/@react-oauth/google.

[33] *Reagraph*. https://github.com/reaviz/reagraph.

[34] *patch-package*. https://www.npmjs.com/package/patch-package.

[35] *ybug-react*. https://ybug.io/.

[36] *highcharts-react-official*. https://www.npmjs.com/package/highcharts-react-official.

[37] *react-toastify*. https://https://www.npmjs.com/package/react-toastify.

[38] *TailwindCSS*. https://tailwindcss.com/.

[39] *DaisyUI*. https://daisyui.com/.