Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Αυτοματοποίηση Λειτουργίας και Documentation API

Μελέτη και υλοποίηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Ναυσικάς Αμπατζή

Επιβλέπων: Βασίλειος Βεσκούκης
Καθηγητής ΕΜΠ

Αθήνα, Ιούνιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Αυτοματοποίηση Λειτουργίας και Documentation API

Μελέτη και υλοποίηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Ναυσικάς Αμπατζή

Επιβλέπων: Βασίλειος Βεσκούκης
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Ιουλίου 2022.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Βασίλειος Βεσκούκης	Γεώργιος Γκούμας	Παναγιώτης Τσανάκας
Καθηγητής ΕΜΠ	Αναπληρωτής Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

Αθήνα, Ιούνιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

(Υπογραφή)

Ναυσικά Αμπατζή
Διπλωματούχα Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Κανείς μπορεί να έχει στα χέρια του το καλύτερο REST API, αλλά θα είναι αδύνατο να το χρησιμοιήσει σωστά εάν πρώτα δεν το έχει κατανοήσει. Σε αυτό το σημείο ορίζεται η έννοια του API Documentation, το οποίο αποτελεί ένα έγγραφο που περιγράφει το πως θα χρησιμοποιηθεί ένα API. Συχνά το Documentation των REST APIs γίνεται με το χέρι, κάτι το οποίο είναι χρονοβόρο και με σημαντικές πιθανότητες λάθους. Υπάρχουν εργαλεία τα οποία διευκολύνουν τη δημιουργία, επεξεργασία και παρουσίαση του API Documentation.

Σε αυτή την εργασία θα παρουσιαστεί μία μέθοδος η οποία συνδυάζει τα εργαλεία Postman και Visual Paradigm, με σκοπό την αυτοματοποίηση της διαδικασίας παραγωγής και παρουσίασης του API Documentation. Μέσω του εργαλείου Postman μπορούν να δημιουργηθούν συλλογές από requests με όλες τις απαραίτητες πληροφορίες, όπως body, responses, headers κ.α. Μέσω του εργαλείου Visual Paradigm μπορεί να αναπαρασταθεί ένα OpenAPI Documentation ή να παραχθεί το Documentation του από μία API αναπαράσταση.

Μέσω της εργασίας αυτής δημιουργήθηκε ένα απλό σύστημα, το οποίο δέχεται ως είσοδο την περιγραφή μίας συλλογής από requests του Postman και παράγει το OpenAPI Documentation. Το αρχείο που παράγεται μπορεί να οπτικοποιηθεί μέσω του Visual Paradigm ως REST API. Επιπλέον το σύστημά μας, επιτελεί και την αντίστροφη λειτουργία. Μέσω ενός REST API σχεδιασμένο στο Visual Paradigm, παράγει το OpenAPI Documentation του. Χρησιμοποιώντας το Documentation αυτό στη συνέχεια μπορεί να δημιουργηθεί ένα REST API στο Postman.

Λέξεις Κλειδιά

Postman, Visual Paradigm, OpenAPI, JSON, API Documentation, Javascript, REST API

Abstract

Someone can have in his hands the best REST API, but it would be impossible to use it correctly if he has not understood it yet. At this point, the concept of API Documentation is defined, which is a document that describes how an API will be used. Often the REST API Documentation is done using a manual approach, which can be time consuming and error-prone. There are tools, that enable the creation, edit and presentation of API Documentation.

In this thesis, I will present a method, that combines the tools Postman and Visual Paradigm to automate the process of production and presentation of API Documentation. Through Postman we can create collections of requests with all the necessary information, such as body, responses, headers etc. Through Visual Paradigm, we can visualize an OpenAPI Documentation or create the Documentation from an API visualization.

Through this thesis, we created a simple system, that gets a description of a collection of requests as input, and generates the OpenAPI Documentation. The file that is generated can be visualized through Visual Paradigm as REST API. Moreover, the system operates and the reverse function. Using a REST API designed in Visual Paradigm, it generates its OpenAPI Documentation. Subsequently, from this Documentation we can generate a REST API in Postman.

Keywords

Postman, Visual Paradigm, OpenAPI, JSON, API Documentation, Javascript, REST API

στους γονείς μου και στον καλύτερό μου φίλο

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Βασίλειο Βεσκούκη για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να ασχοληθώ με το παρόν θέμα.

Επίσης, θα ήθελα να ευχαριστήσω τον Υποψήφιο Διδάκτορα και Ερευνητή Γιάννη Τζαννέτο για τη συμβολή και συνεργασία του.

Ακόμα, οφείλω ένα μεγάλο ευχαριστώ στους φίλους μου για την στήριξη και συνεργασία όλα αυτά τα χρόνια.

Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Θέμα

Ανάμεσα σε όλες τις φάσεις του κύκλου ζωής ενός API, το Documentation πλέον φαίνεται να έχει την μεγαλύτερη ανάπτυξη. Αυτό οφείλεται σίγουρα στην εξέλιξη των Documentation tools, καθώς στο παρελθόν δινόταν αρκετά λιγότερη έμφαση σε αυτό το κομμάτι. Μάλιστα θεωρείται πιο εύκολο κανείς να γράψει καλό κώδικα, παρά Documentation.

Οι λόγοι για να αναπτύξει κανείς καλό Documentation είναι ποικίλοι. Αρχικά, οι developers κατανοούν πιο εύκολα τα services που παρέχονται και τη σημασία τους, οδηγώντας σε πιο εύκολη και σωστή ανάπτυξή τους. Η ευχρηστία αυτή, οδηγεί σε καλύτερη διαφήμιση και αναγνωρισιμότητα του προϊόντος. Επιπλέον, εξοικονομείται χρόνος καθώς η κατανόηση των χαρακτηριστικών είναι πιο εύκολη.

Πλέον υπάρχουν εργαλεία τα οποία μπορούν να παράξουν Documentation από ορισμένες μορφές ενός API ή να αναπαραστήσουν REST APIs γραφικά. Η αφορμή αυτής της εργασίας είναι η σημασία και ανάγκη για αυτοματοποίηση της παραγωγής και κατανόησης του API Documentation μέσω εργαλείων ευρέως χρησιμοποιούμενων, όπως το Postman και το Visual Paradigm.

Αρχικά, μέσω του Postman είναι δυνατόν να δημιουργηθούν APIs από τα οποία μπορούμε να έχουμε Documentation. Επιπλέον, μέσω του εργαλείου Visual Paradigm μπορούμε να οπτικοποιήσουμε ένα REST API είτε δημιουργώντας το, είτε από το OpenAPI Documentation του. Ωστόσο, παρόλο που κάποιος έχει τη δυνατότητα να δημιουργήσει ένα API στο Postman συνήθως δημιουργεί μόνο ένα Collection, το οποίο απέχει πολύ από το να περιέχει το Documentation. Αυτό υπήρξε και το κίνητρο για τη δουλειά μας. Ο στόχος αυτής της εργασίας είναι διπλός. Ο πρώτος είναι η αυτόματη παραγωγή OpenAPI Documentation από συλλογές του Postman με στόχο την οπτικοποίησή του στο εργαλείο Visual Paradigm. Ο δεύτερος είναι η δημιουργία OpenAPI Documentation από ένα REST API το οποίο έχει σχεδιαστεί στο Visual Paradigm και η αξιοποίησή του μέσω του εργαλείου Postman.

Για τους παραπάνω σκοπούς αναπτύχθηκε ένα σύστημα μέσω του οποίου κανείς εισάγει απλά τα Postman Collection αρχεία και γίνεται αυτόματα η μετατροπή τους σε κατάλληλη OpenAPI μορφή. Το αρχείο που προκύπτει οπτικοποιείται μέσω του Visual Paradigm και στη συνέχεια μπορεί να εξαχθεί ώτσε να δημιουργηθεί ένα API.

1.2 Οργάνωση του τόμου

Η εργασία αυτή είναι οργανωμένη σε επτά κεφάλαια:

1. Στο κεφάλαιο 2 παρουσιάζεται το θεωρητικό μέρος των τριών κύριων τεχνολογιών που χρησιμοποιήθηκαν. Αυτές είναι το OpenAPI, το Postman και το Visual Paradigm.
2. Στο κεφάλαιο 3 παρουσιάζεται το σύστημα που δημιουργήθηκε, ώστε από ένα Collection του εργαλείου Postman να έχουμε ένα οπτικοποιημένο Documentation του REST API στο Visual Paradigm.
3. Στο κεφάλαιο 4 παρουσιάζεται το σύστημα που αναπτύχθηκε, ώστε από ένα REST API σχεδιασμένο στο Visual Paradigm να έχουμε το OpenAPI Documentation του, το οποίο θα μπορεί να φορτωθεί στο Postman και να δημιουργήσει ένα νέο API.
4. Στο κεφάλαιο 5 παρουσιάζεται ο τρόπος χρήσης του συστήματός μέσω ενός Command Line Interface.
5. Στο κεφάλαιο 6 παρουσιάζεται ο τρόπος χρήσης του συστήματός μέσω ενός απλού Web Application.
6. Στο κεφάλαιο 7 γίνεται μία σύντομη αναφορά στα συμπεράσματα και σε μελλοντικές επεκτάσεις αυτής της εργασίας.

Μέρος I

APIs και Εργαλεία Σχεδίασης, Ανάπτυξης και Τεκμηρίωσης

Κεφάλαιο 2

OpenAPI και Εργαλεία

Στο κεφάλαιο αυτό παρουσιάζονται αναλυτικά οι τρεις βασικές τεχνολογίες που έχουν σχέση με την εργασία αυτή, δηλαδή το OpenAPI 3.0, το Postman και το Visual Paradigm. Επιπλέον, θα παρουσιαστεί το εργαλείο Swagger.

2.1 OpenAPI

Πριν προχωρήσουμε στην ανάλυση του OpenApi θα δωθεί ένας ορισμός του API [1]. Το API είναι η συντομογραφία του Automatic Programming Interface και αποτελείται από μια σειρά κανόνων (rules). Τα APIs επιτρέπουν σε ένα application να εξάγει πληροφορίες από ένα κομμάτι λογισμικού και να χρησιμοποιήσει αυτές τις πληροφορίες σε μια άλλη εφαρμογή, ή ορισμένες φορές για ανάλυση δεδομένων. Με απλούς όρους, ένα API είναι ένα σχέδιο (blueprint) που επιτρέπει σε εφαρμογές να μιλάνε και να συνεργάζονται με άλλες εφαρμογές (και όχι μόνο), να μιλάνε δηλαδή μεταξύ τους.

Το OpenAPI Specification [2], αποτελεί μία open source προδιαγραφή για περιγραφή και documentation των APIs. Η τελευταία έκδοση του OpenAPI είναι η 3.0 και μπορεί να γραφτεί είτε σε JSON είτε σε YAML μορφή. Στη συνέχεια για την ανάλυσή του θα χρησιμοποιηθεί η YAML μορφή, καθώς είναι πιο ευανάγνωστη.
Ένα OpenAPI Specification έχει την εξής μορφή:

```
openapi: 3.0.0
info:
  version: 1.0.0
  title: Sample API
  description: A sample API to illustrate OpenAPI concepts
paths:
  /list:
    get:
      description: Returns a list of stuff
      responses:
        '200':
          description: Successful response
```

Σχήμα 2.1: *OpenAPI Format*

Ένα API που ορίζεται μέσω του OpenAPI μπορεί να χωριστεί σε τρεις κατηγορίες:

1. Meta Information
2. Endpoints: Παράμετροι, Request Bodies, Responses
3. Reusable Components: Schemas, Responses, Παράμετροι, Άλλα Components

2.1.1 Meta Information

Ακολουθεί ένας απλός ορισμός ενός API, που περιέχει meta information, όπως ο τίτλος (title), η έκδοση (version) και το server url.

```
openapi: 3.0.0
info:
  version: 1.0.0
  title: Simple Artist API
  description: A simple API to illustrate OpenAPI concepts

servers:
  - url: https://example.io/v1

# Basic authentication
components:
  securitySchemes:
    BasicAuth:
      type: http
      scheme: basic
security:
  - BasicAuth: []

paths: {}
```

Σχήμα 2.2: *OpenAPI Meta Information*

2.1.2 Path Items

Τα Path Items αποτελούν τα endpoints του API και είναι σχετικά με το server url.

```
paths:
  /artists:
    get:
     description: Returns a list of artists
```

Σχήμα 2.3: *OpenAPI Paths*

2.1.3 Responses

Το response ορίζεται από το HTTP status code και τα δεδομένα που επιστρέφονται (body ή/και headers). Το description δίνει πληροφορίες για το response.

```
responses:
  '200':
    description: Successfully returned a list of artists
    content:
      application/json:
        schema:
          type: array
          items:
            type: object
            required:
              - username
```

Σχήμα 2.4: *OpenAPI Response*

2.1.4 Παράμετροι

Τα RESTful parameters ορίζουν τις μεταβλητές που χρησιμοποιούνται.

2.1.4.1 Query Parameters

Τα Query Parameters εμφανίζονται στο τέλος του URL με ένα ? στο τέλος τους. Είναι προαιρετικά και όχι απαραίτητα μοναδικά.

2.1.4.2 Request Body

Τα αιτήματα POST, PUT, και PATCH συνήθως περιέχουν ένα Request Body, το οποίο ορίζεται κάτω από το πεδίο requestBody.

```
post:
  description: Lets a user post a new artist
  requestBody:
    required: true
    content:
      application/json:
        schema:
          type: object
          required:
            - username
          properties:
            artist_name:
              type: string
```

Σχήμα 2.5: *OpenAPI Request Body*

2.1.4.3 Path Parameters

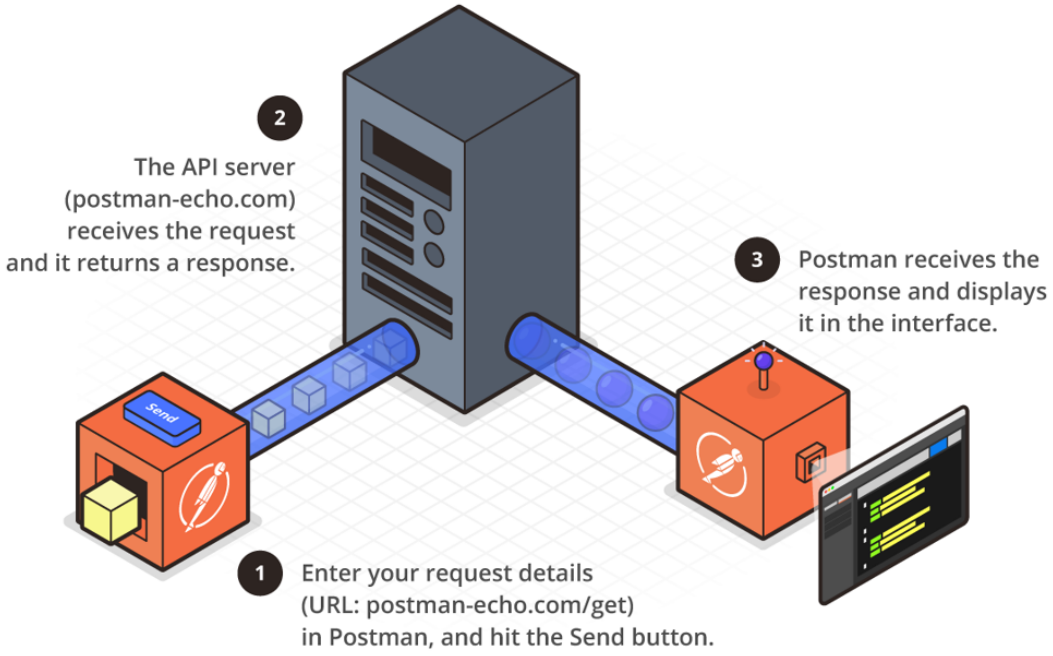
Τα Path Parameters αποτελούν μέρος του URL και απομονώνουν ένα μέρος των δεδομένων που στέλνονται μέσω του URL.

2.2 Postman

Το Postman [3] αποτελεί ένα εργαλείο που χρησιμοποιείται στην ανάπτυξη Rest APIs. Προσφέρει έναν εύκολο τρόπο για να δοκιμαστεί ένα Rest API στέλνοντας HTTP Request [4] από το γραφικό του περιβάλλον.

2.2.1 Αποστολή Requests

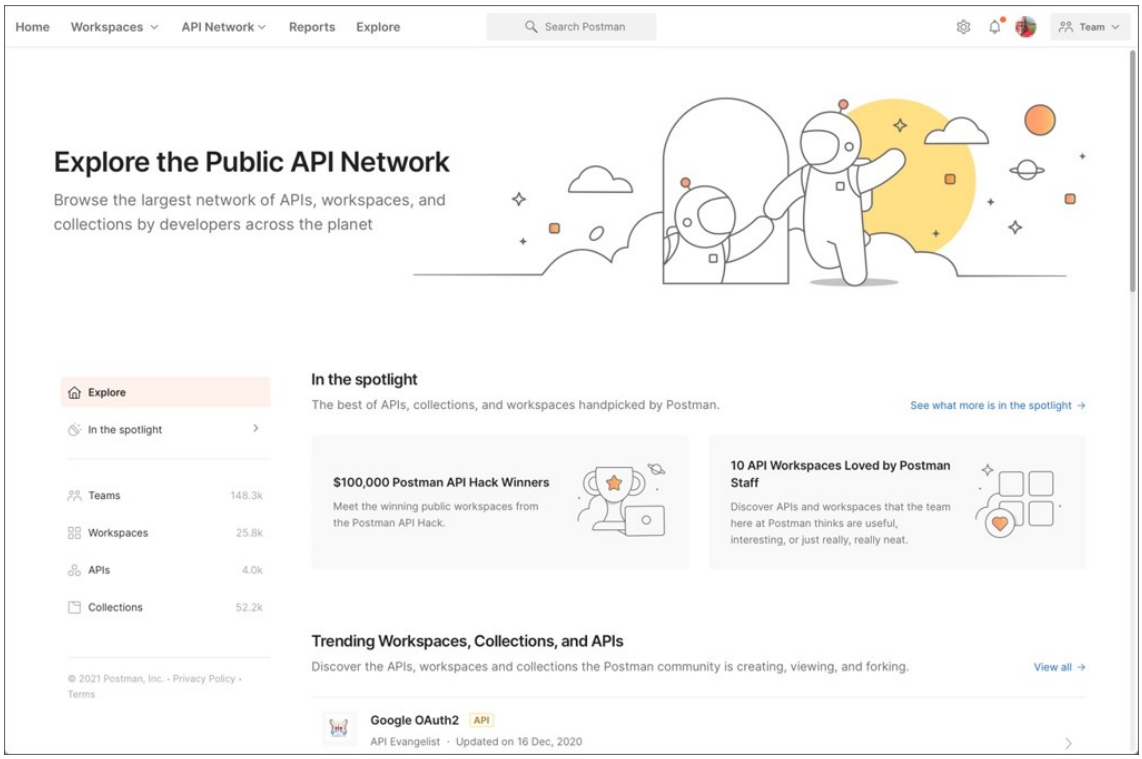
Η αποστολή των Requests όπως φαίνεται και στην εικόνα 2.6, γίνεται σε τρία βήματα. Αρχικά, ο χρήστης στέλνει το HTTP Request από το γραφικό περιβάλλον του Postman. Στη συνέχεια, το λαμβάνει ο server του Postman (postmanecho.com) και επιστρέφει μία απάντηση. Τέλος, η απάντηση αυτή λαμβάνεται από το Postman και απεικονίζεται στο γραφικό του περιβάλλον.



Σχήμα 2.6: *Postman Requests*

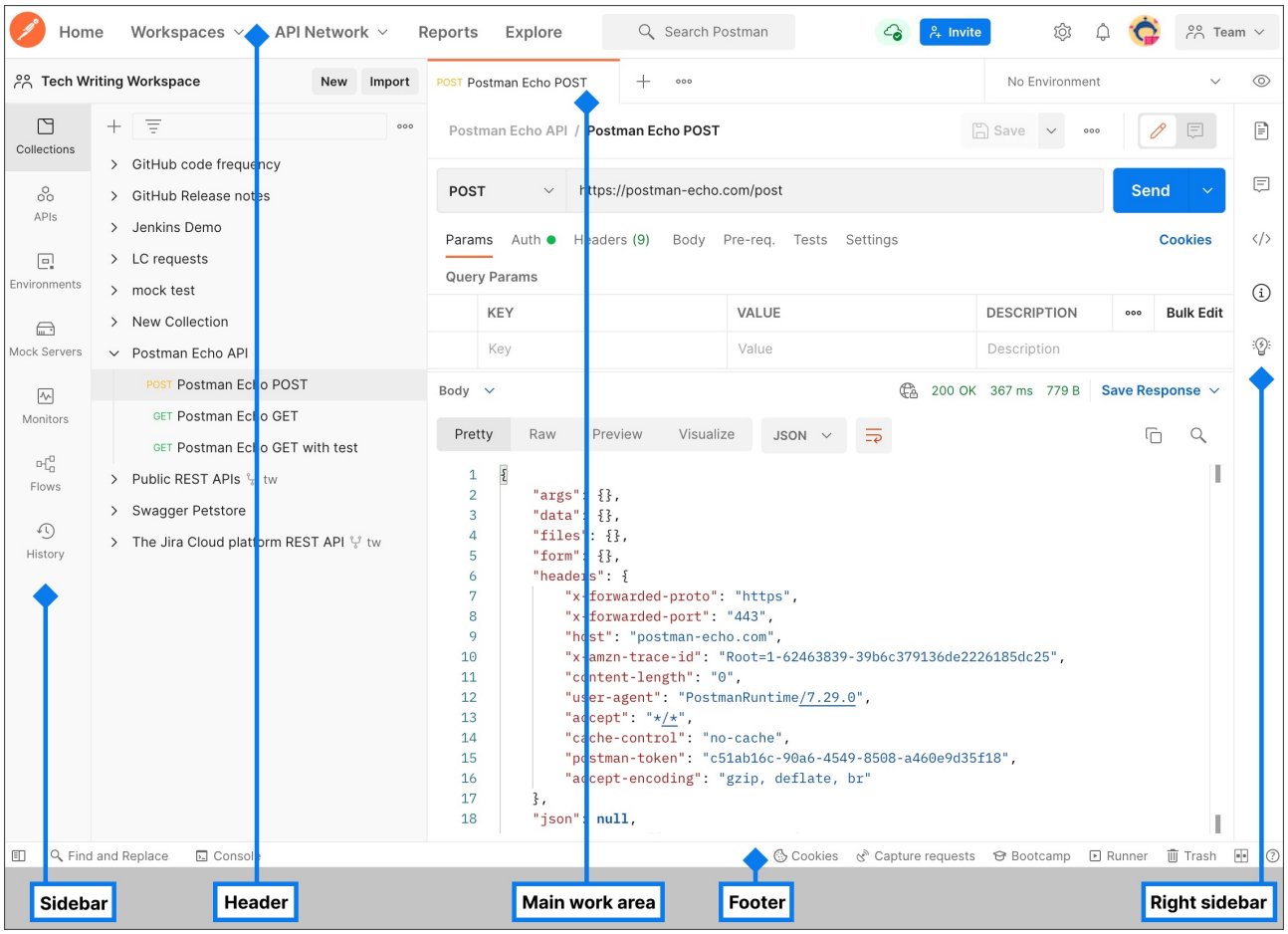
2.2.2 Public API

Το μεγαλύτερο δίκτυο από APIs, collections και workspaces των developers.



Σχήμα 2.7: *Postman Public API*

2.2.3 Περιβάλλον του Postman



Σχήμα 2.8: *Postman*

2.2.3.1 Sidebar

Το Sidebar παρέχει πρόσβαση στις βασικές λειτουργίες του Postman.

1. History: Παρουσιάζεται το ιστορικό των requests.
2. Collections: Group από αποθηκευμένα requests.
3. APIs: Δυνατότητα για δημιουργία ενός API από την αρχή, testing, versioning και γενικότερα τη διαχειρισή του.
4. Environments: Το environment είναι ένα set από variables που χρησιμοποιούνται στα requests.
5. Mock Servers: Οι Mock Servers προσομοιώνουν τη λειτουργία ενός API, επιστρέφοντας ήδη ορισμένα data, δίνοντας τη δυνατότητα για ανάπτυξη και testing του API. Βασίζονται σε examples που είναι αποθηκευμένα σε ένα Collection για να επιστρέψουν mock data.

6. Monitors: Παρέχεται η δυνατότητα εκτέλλεσης των Tests με αυτοματοποιημένο και χρονικά προσδιορισμένο τρόπο.

2.2.3.2 Header

Οι κύριες δυνατότητες που παρέχονται μέσω του Header είναι η δημιουργία workspaces, η πρόσβαση σε repots των APIs, η εξερεύνηση του Public API Network, η αναζήτηση στο Postman και η πρόσβαση στις ρυθμίσεις. Μέσω των Workspaces οργανώνονται τα APIs και διευκολύνεται η συνεργασία ανάμεσα σε ομάδες. Το Public API Network αποτελεί το μεγαλύτερο δίκτυο από APIs, collections και workspaces των developers.

2.2.3.3 Main Work Area

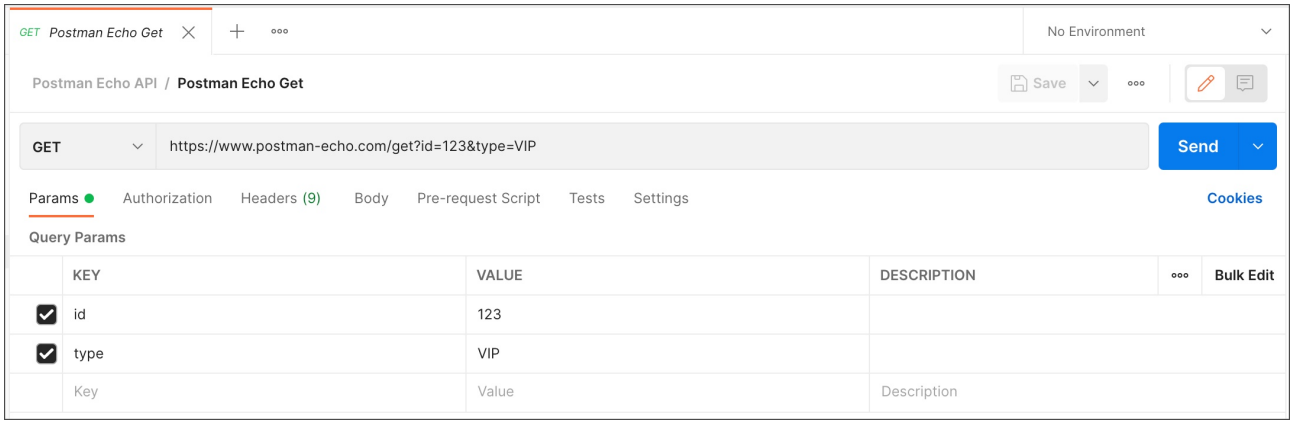
Μέσω των Tabs μπορούμε να πλοηγούμαστε και να οργανώνουμε δραστηριότητες των Collections, των APIs κ.α., όπως αποστολή requests, αποθήκευση αλλαγών και προβολή Documentation.

2.2.3.4 Right Sidebar

Μέσω του Right Sidebar δίνεται πρόσβαση σε εργαλεία όπως documentation, σχόλια και πληροφορίες των requests.

2.2.4 Postman Requests

Το Postman δίνει τη δυνατότητα για αποστολή requests για σύνδεση με APIs. Τα requests μπορούν να ανακτήσουν, προσθέσουν, διαγράψουν και ενημερώσουν δεδομένα. Μέσω αυτών μπορούν να σταλούν παράμετροι, λεπτομέρειες ταυτοποίησης και body data. Κατά την αποστολή πρέπει να εισαχθεί το λιγότερο το URL και η μέθοδος.

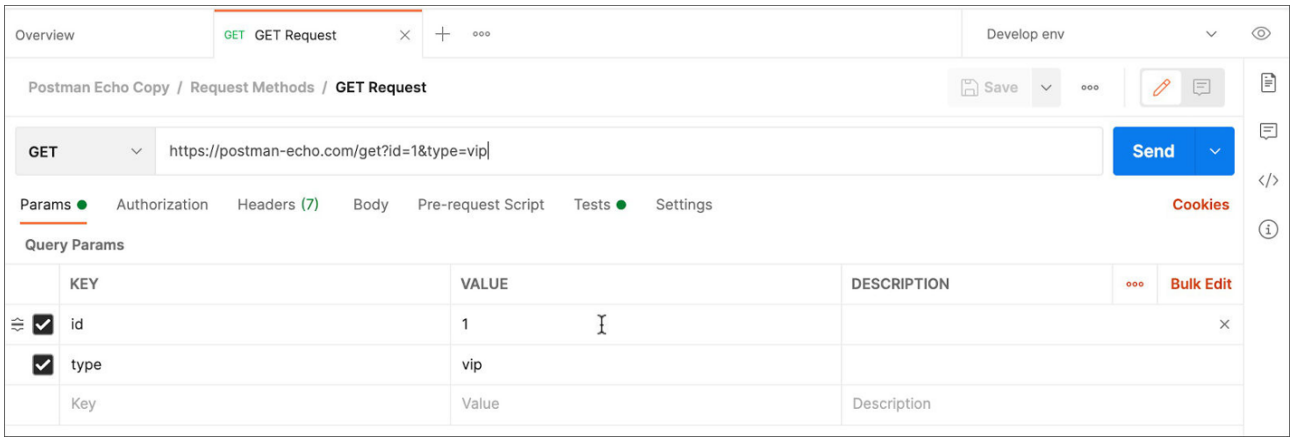


Σχήμα 2.9: *Postman Request*

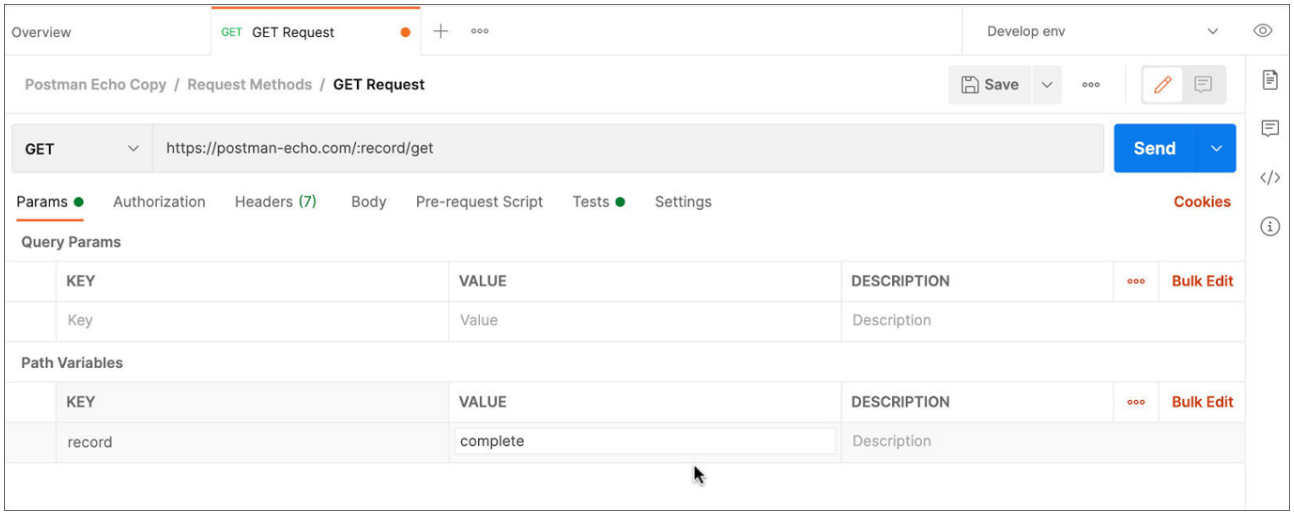
Όπως φαίνεται και στην εικόνα 2.9 ο χρήστης ορίζει τη μέθοδο του request, το url και διάφορες άλλες λεπτομέρειες, όπως οι παράμετροι, το body και τα headers.

2.2.4.1 Αποστολή Παραμέτρων

Οι παράμετροι μπορεί να είναι είτε query, είτε path και αποστέλλονται είτε μέσω του URL είτε μέσω συμπλήρωσης πεδίων στο equest. Τα query parameters είναι παράμετροι που στέλνονται μέσω του URL. Εισάγονται στο τέλος του και ορίζουν συγκεκριμένο περιεχόμενο ή ενέργειες σύμφωνα με τα δεδομένα που αποστέλλονται. Τα path parameters είναι μεταβλητές που ¨δείχνουν¨ σε συγκεκριμένο resource μέσα σε ένα collection. Μέσω του Bulk Edit, συμπληρώνονται ως text, αντί για χρήση του UI.



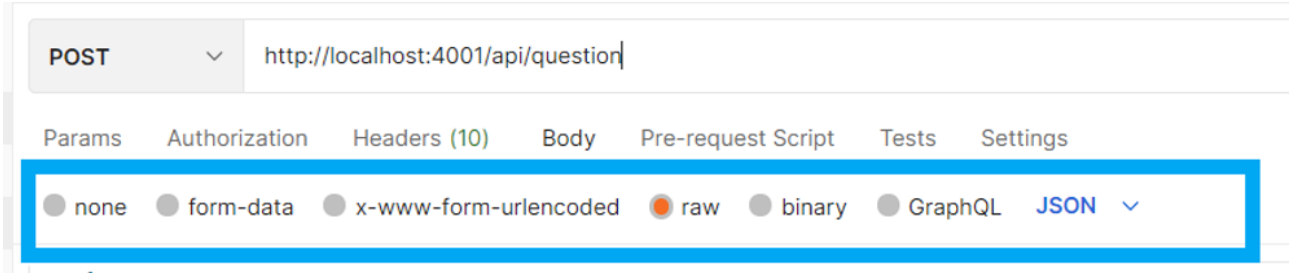
Σχήμα 2.10: *Postman Query Parameters*



Σχήμα 2.11: *Postman Path Parameters*

2.2.4.2 Body Data

Τα body δεδομένα αποστέλλονται μέσω PUT, POST και PATCH requests. Πρέπει να ελέγχονται και τα headers.

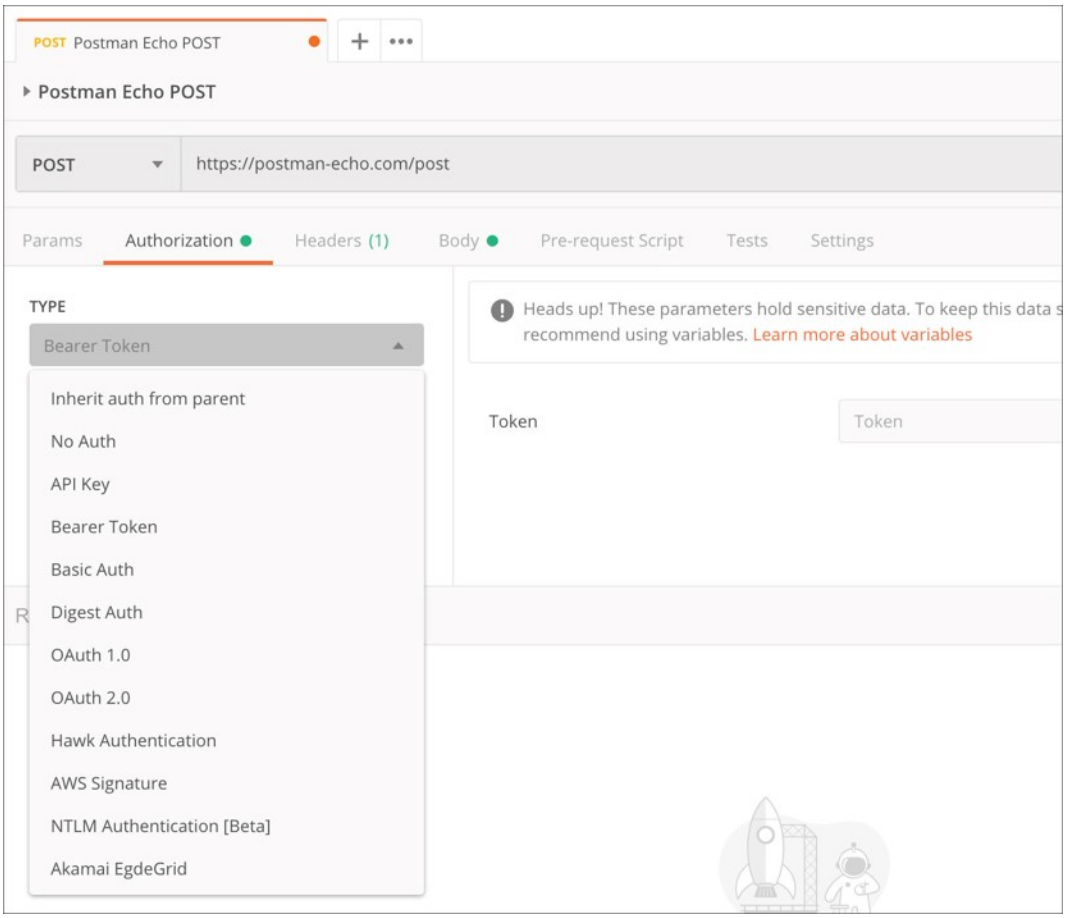


Σχήμα 2.12: *Postman Body Data*

2.2.4.3 Authorizing Requests

- Τα APIs χρησιμοποιούν authorization ώστε τα αιτήματα έχουν πρόσβαση στα δεδομένα με ασφαλή τρόπο. Αυτό μπορεί να περιλαμβάνει authentication του αποστολέα, ώστε να γίνει γνωστό εάν αυτός έχει το δικαίωμα επεξεργασίας των δεδομένων.
- Τα auth δεδομένα μπορεί να εισαχθούν στην επικεφαλίδα, στο body ή στις παραμέτρους στο request.
- Είναι δυνατό να εισαχθούν auth data και σε ολόκληρο το Collection. Τα δεδομένα αυτά θα ενσωματωθούν και στα επιμέρους requests που διαθέτει.

Μέσω του tab Authorization της εικόνας 2.13 επιλέγουμε τον τύπο και συμπληρώνουμε τα ανάλογα πεδία. Το Postman θα τα εμφανίσει στα αντίστοιχα πεδία του request.



Σχήμα 2.13: *Postman Authorization Tab*

2.2.5 Postman Responses

Μέσω του Postman Response Viewer μπορεί να οπτικοποιείται και να ελέγχεται η ορθότητα ενός response. Ένα API response αποτελείται από το body,τα headers και το status code και μπορεί να αποθηκευτεί. Επιπλέον, ένα response μπορεί να αποθηκευτεί ως example και να χρησιμοποιηθεί στη συνέχεια για testing. Το Postman δίνει τη δυνατότητα για αναπαράσταση των responses με χρήση HTML και CSS [5].

2.2.6 Examples

- Τα Examples είναι ένα «ζευγάρι» request και response. Είναι δυνατό να δημιουργηθεί πριν την αποστολή του «κανονικού» request.
- Διευκολύνουν στην κατανόηση του API, ειδικά εάν το endpoint δεν έχει υλοποιηθεί ακόμα ή ο server δεν είναι έτοιμος.
- Μέσω του Postman mock server, μπορεί να δημιουργηθεί ένα mock endpoint με mock responses. Έτσι μπορούν να γίνονται requests με σκοπό τη δημιουργία του frontend ή test scripts. Προσομοιώνεται δηλαδή η συμπεριφορά του API.
- Είναι δυνατό να αποθηκευτεί ως example και ένα request και response από server.

2.2.7 API Development

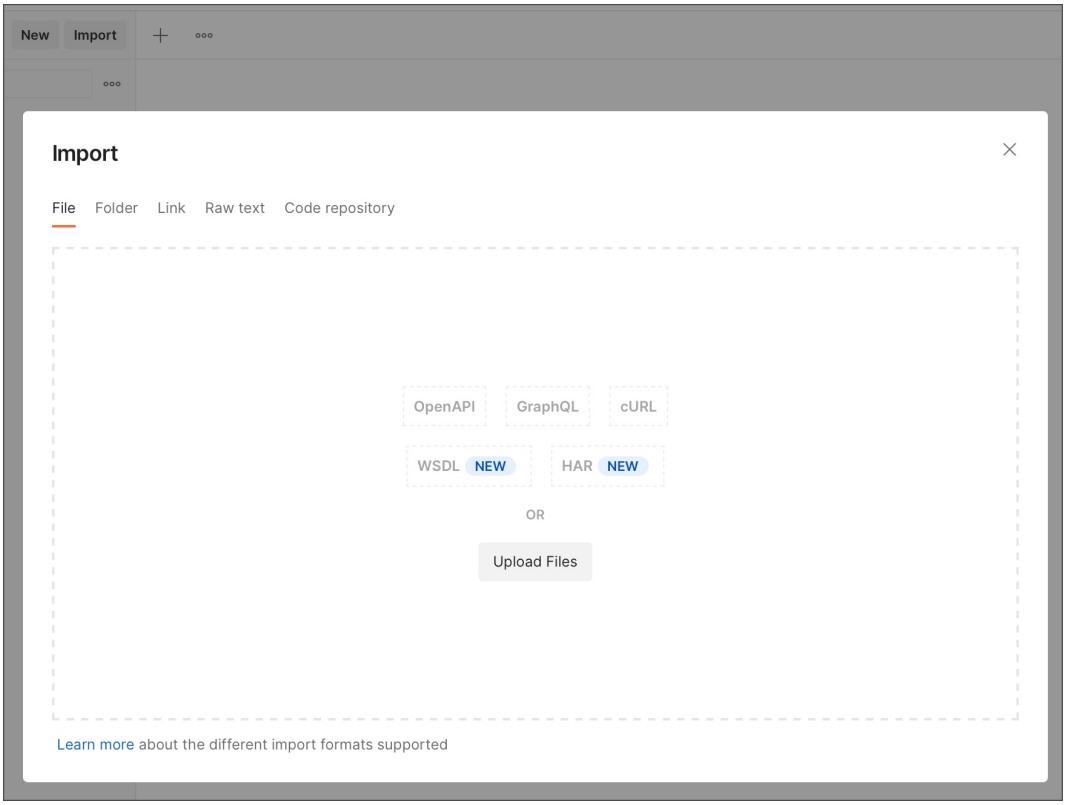
Μέσω του Postman είναι δυνατή η δημιουργία και διαχείριση APIs μέσω του Tool API Builder [6]. Ακολουθεί μία συνοπτική περιγραφή των βασικών δυνατοτήτων του:

- Δημιουργία, μετονομασία και διαγραφή APIs.
- Versioning: Κάθε API έχει μία ή περισσότερες εκδόσεις και οι εκδόσεις έχουν πολλαπλά releases. Επίσης κάθε έκδοση έχει ένα status.
- API Specification: Δυνατότητα για επεξεργασία του API schema και δημιουργίας Collection από αυτό.
- Developing: Προσθήκη Documentation, environments, mock server [7].
- Testing: Δημιουργία collections και σύνδεση τους στο API για testing.
- Monitors: Μέσω αυτών καταγράφεται η λειτουργικότητα, η απόδοση και ο χρόνος απόκρισης του API.

2.2.8 Εισαγωγή και Εξαγωγή δεδομένων

2.2.8.1 Εισαγωγή Δεδομένων

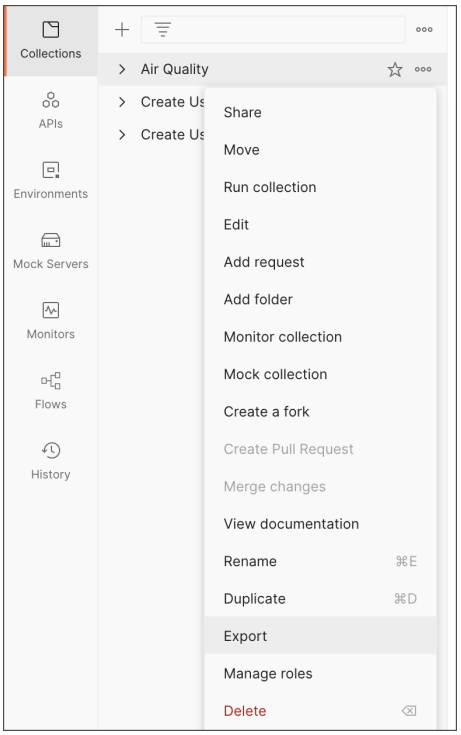
Μέσω του Postman είναι δυνατή η άμεση εισαγωγή Collections ή API specifications. Για την εισαγωγή δεδομένων επιλέγεται το Import, όπως φαίνεται παρακάτω. Υπάρχει η δυνατότητα να εισαχθούν δεδομένα από αρχεία, φακέλους, links, απλού κειμένου ή code repositories.



Σχήμα 2.14: *Postman Data Import*

2.2.8.2 Εξαγωγή Δεδομένων

Μέσω του Postman είναι δυνατή η εξαγωγή δεδομένων, όπως collections, environments, data dumps, και globals, όπως JSON αρχεία. Θα παρουσιαστεί εκτενέστερα η εξαγωγή Collections, καθώς σε αυτό εστιάζει το σύστημα που αναπτύχθηκε. Όπως φαίνεται στην εικόνα 2.15 εάν επιλέξουμε το κουμπί more actions, θα εμφανιστεί μεταξύ άλλων η επιλογή Export. Το αρχείο που προκύπτει είναι σε μορφή json.



Σχήμα 2.15: *Postman Export Collection*

2.3 Visual Paradigm

Το Visual Paradigm [8] αποτελεί ένα software application που σχεδιάστηκε για software development ομάδες με σκοπό τη μοντελοποίηση business information συστημάτων και τη διαχείριση διαδικασιών σχετικών με το development. Επιπλέον, προσφέρει τη δυνατότητα παραγωγής reports, αλλά και code engineering λειουργίες, όπως η παραγωγή κώδικα. Μπορεί να χρησιμοποιηθεί τοπικά ως εφαρμογή, αλλά και ως online tool. Όσες διαδικασίες περιγραφούν παρακάτω αφορούν την τοπική εφαρμογή, καθώς στο online tool δεν είνα διαθέσιμες. Όσον αφορά τα διαγράμματα, υποστηρίζει 14 τύπους, όπως Class [9], Use Case [10], Sequence [11] διαγράμματα.
Η παρούσα διπλωματική θα αξιοποιήσει τις δυνατότητες Generate OpenAPI και Reverse OpenAPI για τις οποίες είναι απαραίτητη η ύπαρξη ενός Class Diagram. Έτσι, η παρουσίαση του Visual Paradigm θα περιοριστεί στην ανάλυση των τριών αυτών εννοιών.

2.3.1 Class Diagram

Τα διαγράμματα κλάσεων περιγράφουν τις οντότητες που απαρτίζουν ένα σύστημα και τις στατικές συσχετίσεις μεταξύ τους [12]. Αποτελούνται από κλάσεις και σχέσεις.

2.3.1.1 Κλάσεις

Οι κλάσεις απεικονίζονται σε Όνομα, Ιδιότητες και Μεθόδους. Οι μέθοδοι χαρακτηρίζονται από την ορατότητά τους όπως και οι ιδιότητες.

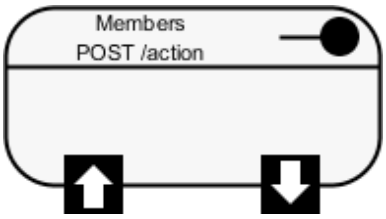
2.3.1.2 Σχέσεις

Οι σχέσεις συνδέουν μεταξύ τους τις κλάσεις ενός διαγράμματος. Η UML ορίζει τις εξής βασικές σχέσεις:

- Εξάρτηση, η οποία δείχνει ότι μία αλλαγή σε μία κλάση επηρεάζει μία άλλη κλάση.
- Γενίκευση (generalisation), η οποία δείχνει ότι μια κλάση είναι ένας πιο εξειδικευμένος τύπος μιας άλλης κλάσης.
- Σύνδεση (association), η οποία δείχνει ότι μία κλάση έχει μία δομική σύνδεση με μία άλλη.
- Υλοποίηση (realisation), η οποία δείχνει ότι μία κλάση υλοποιεί κάποια προδιαγραφή (specification). Η προδιαγραφή συνήθως είναι μια διεπαφή (interface).

2.3.2 Σχεδιασμός REST API με UML

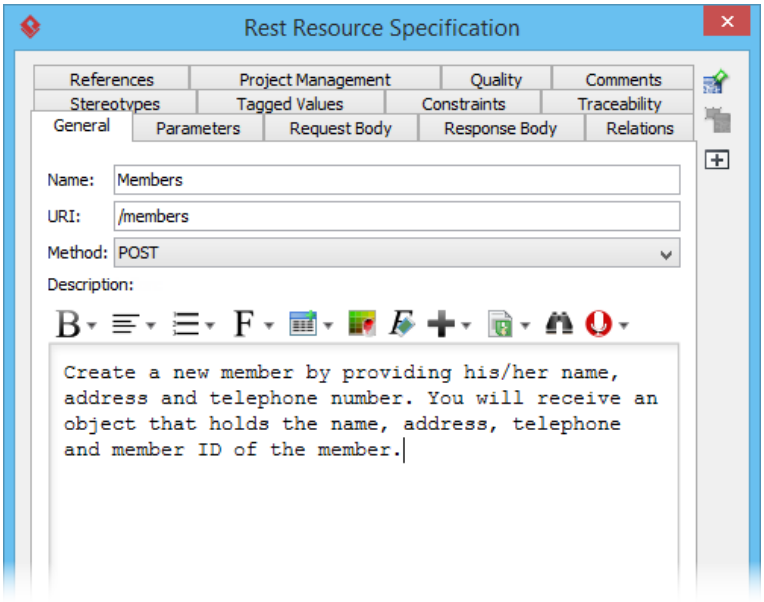
Μέσω του Visual Paradigm μπορεί κανείς να σχεδιάσει ένα REST API, χρησιμοποιώντας ένα Class Diagram, το οποίο θα αναπαριστά το resource, το request και το response body. Στο toolbar του Class Diagram επιλέγουμε το REST Resource :



Σχήμα 2.16: *Visual Paradigm Rest Resource*

2.3.2.1 Specification

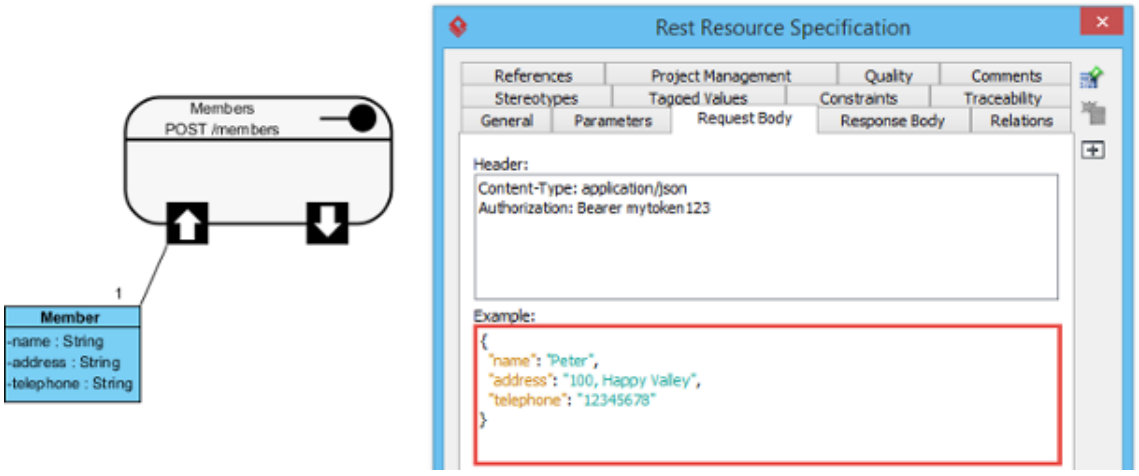
Μπορούμε να ορίσουμε τα πεδία του Specification, όπως φαίνεται παρακάτω. Τα κύρια πεδία είναι το URI και η μέθοδος.



Σχήμα 2.17: *Visual Paradigm Rest Specification*

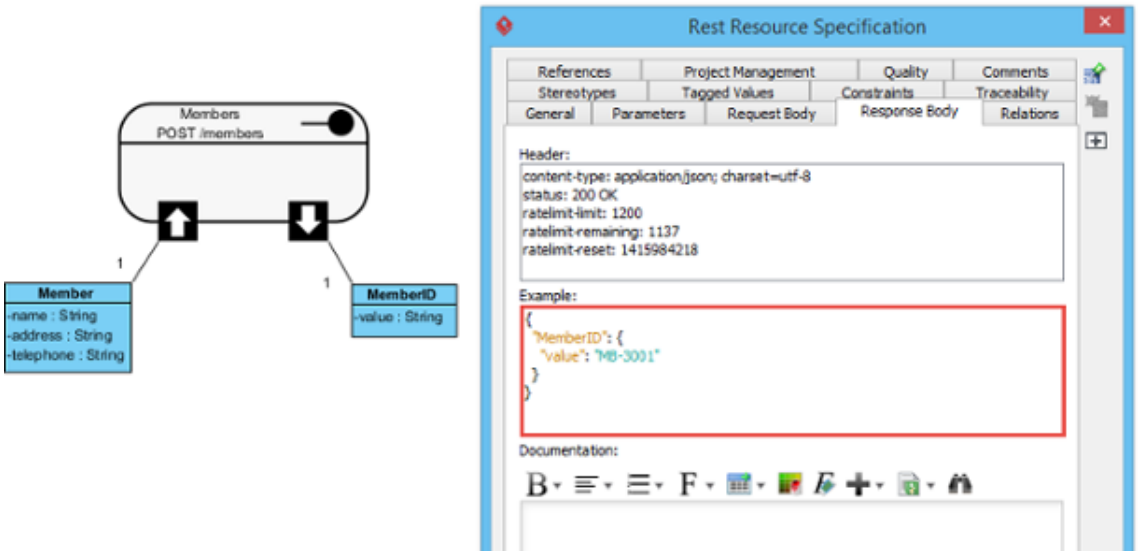
2.3.2.2 Request, Response Body

Εάν η μέθοδος είναι POST, PUT, PATCH ή DELETE και υπάρχει Request Body, αυτό μπορεί να αναπαρασταθεί με μία κλάση, η οποία θα περιέχει και τις παραμέτρους του Request Body. Ακολουθεί η παρουσίαση του Request Body ως κλάση συνδεδεμένη στο Rest API και όπως απεικονίζεται στο Specification του:



Σχήμα 2.18: *Visual Paradigm Rest Request Body*

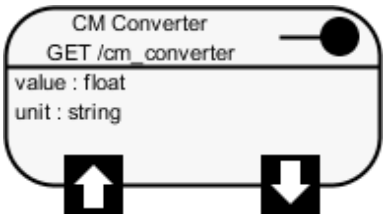
Η ίδια απεικόνιση υπάρχει και στην περίπτωση ενός response ενός REST API, όπως φαίνεται παρακάτω:



Σχήμα 2.19: *Visual Paradigm Rest Response Body*

2.3.2.3 Parameters

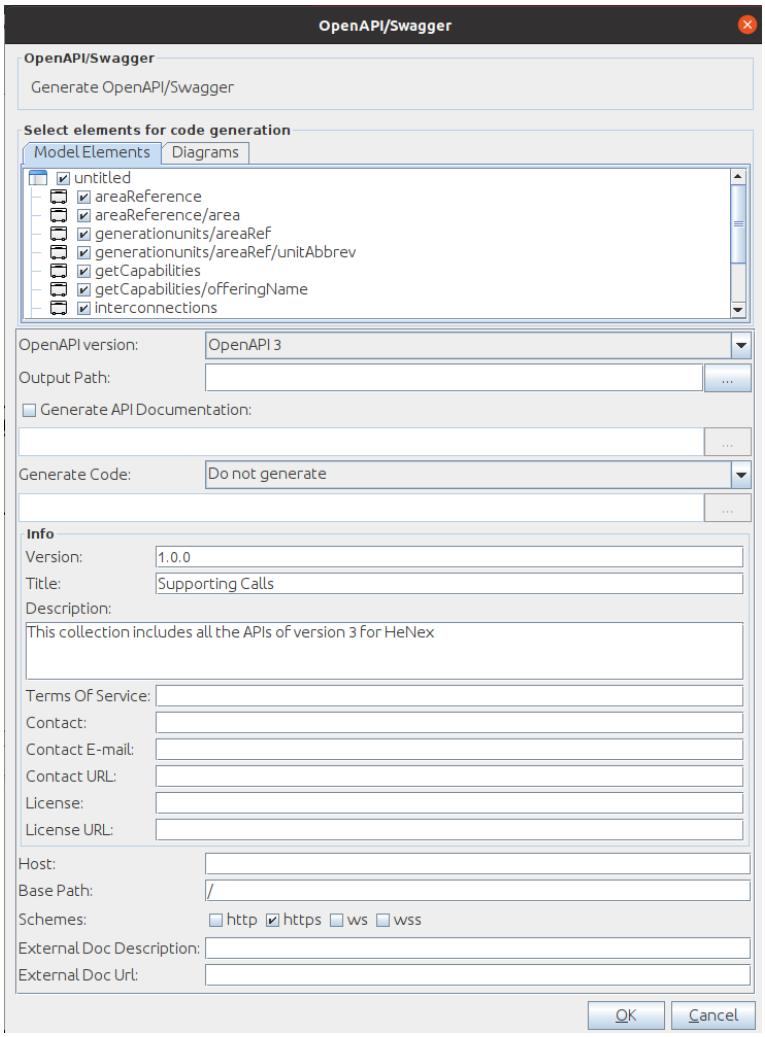
Τα query parameters, τα οποία στέλνουν πληροφορίες στο service είναι μη υποχρεωτικά και μπορούν να αναπαρασταθύν ως εξής:



Σχήμα 2.20: *Visual Paradigm Rest Parameters*

2.3.3 Παραγωγή REST API από UML

Αφού έχει σχεδιαστεί το REST API μέσω του class διαγράμματος, μπορεί να παραχθεί σε μορφή OpenAPI 3/Swagger2.



Σχήμα 2.21: *Visual Paradigm Export REST API*

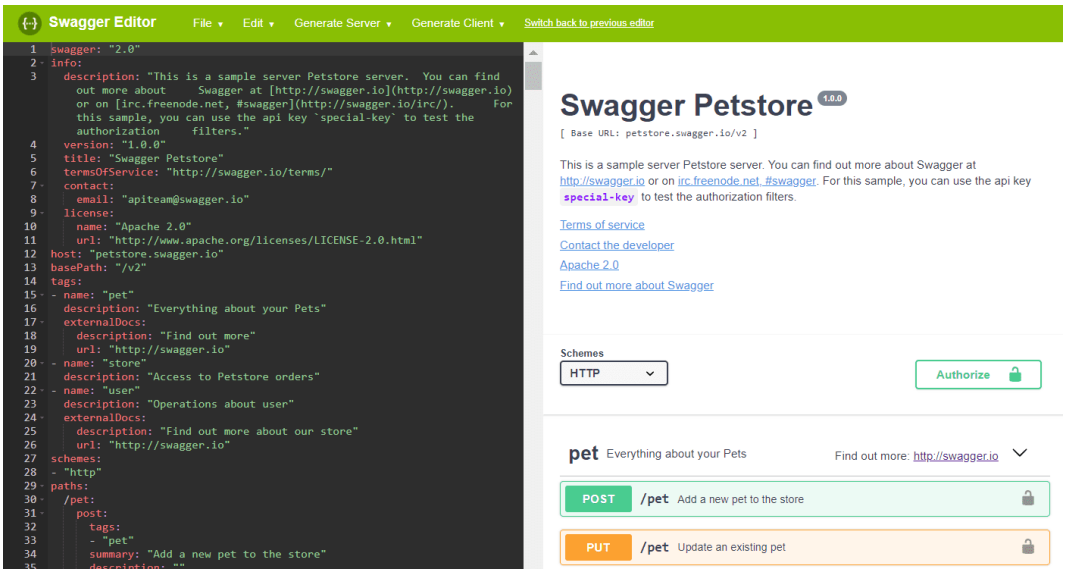
2.3.4 Reverse OpenAPI/Swagger

Στο Visual Paradigm υπάρχει η δυνατότητα του Reverse OpenAPI/Swagger, δηλαδή η εισαγωγή ενός αρχείου σε μία από τις δύο μορφές και η απεικόνιση του ως REST API μέσω ένος Class διαγράμματος.

2.4 Swagger Tool

Το Swagger [13] αποτελεί το πιο γνωστό και ευρέως χρησιμοποιούμενο εργαλείο για OpenAPI Specification. Περιέχει ένα συνδυασμό από δωρεάν open source και commercial εργαλεία, τα οποία μπορούν να χρησιμοποιηθούν σε διάφορα στάδια του API Lifecycle. Ο χρήστης μπορεί να καταλάβει και να αλληλεπιδράσει με το API με λίγη υλοποίηση λογικής. Παρέχει μία αρκετά διαδραστική επαφή, ώστε οι χρήστες να κατανοούν και να δοκιμάζουν το API γρήγορα. Ακολουθούν τα βασικά του εργαλεία:

- Swagger Specification: Τρόπος περιγραφής ενός API. Περιέχει πληροφορίες όπως οι παράμετροι, τα αποτελέσματα, απαιτήσεις πιστοποίησης, endpoints κα.
- Swagger Editor: OpenSource Editor για σχεδιασμό, ορισμό και documentation των APIs. Ο browser-based editor ¨επιστρέφει¨ OpenAPI Specifications, εμφανίζει errors και παρέχει real-time feedback.
- Swagger User Interface: Δίνει τη δυνατότητα παραγωγής API Documentation



Σχήμα 2.22: *Swagger Editor*

Μέρος II

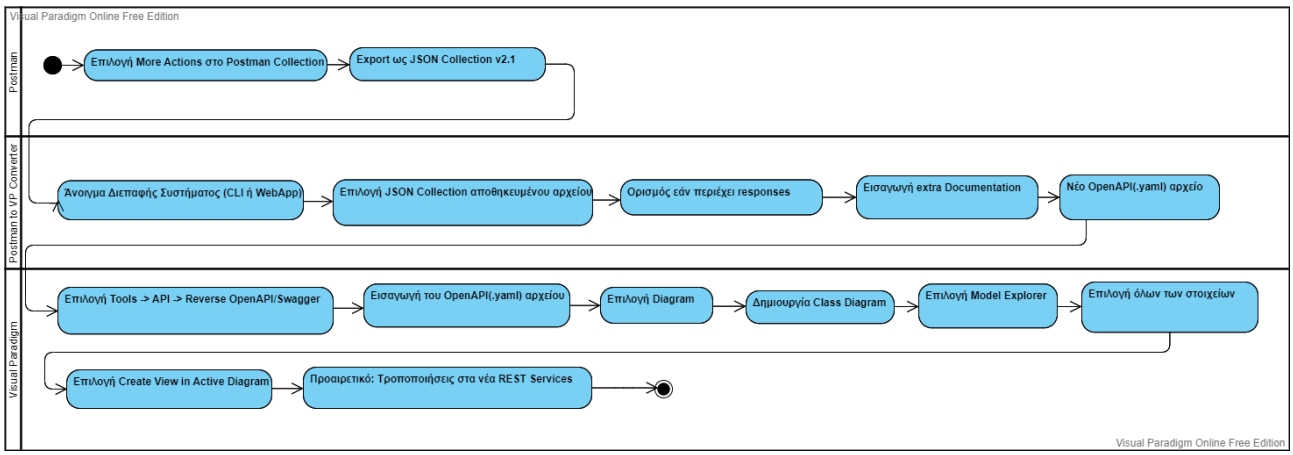
Ανάπτυξη Ενός Εργαλείου

Κεφάλαιο 3

Postman Collection Documentation to REST API Visual Paradigm

Στο κεφάλαιο αυτό παρουσιάζεται η μελέτη που έγινε για την υλοποίηση του συστήματος, μέσω του οποίου απεικονίζουμε ένα Postman Collection Documentation σε REST API στο Visual Paradigm. Όπως αναφέρθηκε και στο θεωρητικό μέρος, το Postman Collection Documentation μπορεί να εξαχθεί σε μορφή JSON μέσω του Postman, ενώ το Visual Paradigm απεικονίζει σε διάγραμμα κλάσεων ένα REST API το οποίο μπορεί να εισαχθεί με τη μορφή OpenAPI (.yaml). Επομένως, για να επιτευχθεί η απεικόνιση, το Postman Collection πρέπει να μετατραπεί σε OpenAPI μορφή. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την ανάλυση που ακολουθεί είναι η Javascript [14].

Ακολουθεί η σχηματική περιγραφή της διαδικασίας που ακολουθεί ο χρήστης κατά τη χρήση του συστήματός μας. Θεωρούμε δεδομένο ότι η διεπαφή (CLI και WebApp) έχει δημιουργηθεί ήδη, καθώς αυτό θα αναλυθεί σε επόμενη ενότητα.



Σχήμα 3.1: *Total System Usage*

Όπως φαίνεται και στην παραπάνω εικόνα 3.1, η διαδικασία περιλαμβάνει τρία εργαλεία, το Postman, το Visual Paradigm και το σύστημα που υλοποιήθηκε (Postman to VP Converter). Ακολουθεί η συνοπτική περιγραφή της παραπάνω διαδικασίας.

1. Αρχικά ο χρήστης εξάγει από το **Postman** το επιλεγμένο Collection σε JSON μορφή.

2. Στη συνέχεια ανοίγει είτε το CLI είτε το Web App του **συστήματος** που δημιουργήσαμε.
3. Επιλέγει το JSON Collection αρχείο.
4. Ορίζει την ύπαρξη ή μη responses στα REST APIs που περιέχει.
5. Προσθέτει τυχόν extra επιθυμητό Documentation το οποίο θα ορίζεται σε όλα τα REST APIs.
6. Δημιουργείται το αρχείο OpenAPI (.yaml) που θα εισαγχθεί στο Visual Paradigm.
7. Πλοηγείται στο **Visual Paradigm** στο Tools − > Reverse OpenAPI Swagger και εισάγει το νέο αρχείο.
8. Δημιουργεί ένα νέο Class Diagram.
9. Επιλέγει όλα τα στοιχεία από το Model Explorer και επιλέγει Create View in Active Diagram.
10. Πλέον μπορεί να δει το οπτικοποιημένο Documentation των REST APIs του Postman Collection.

Ακολουθεί η διαδικασία της δημιουργίας του συστήματός μετατροπής του Postman Collection σε OpenAPI μορφή κατάλληλη για οπτικοποίηση μέσω του Visual Paradigm.

3.1 Υφιστάμενο Υλικό

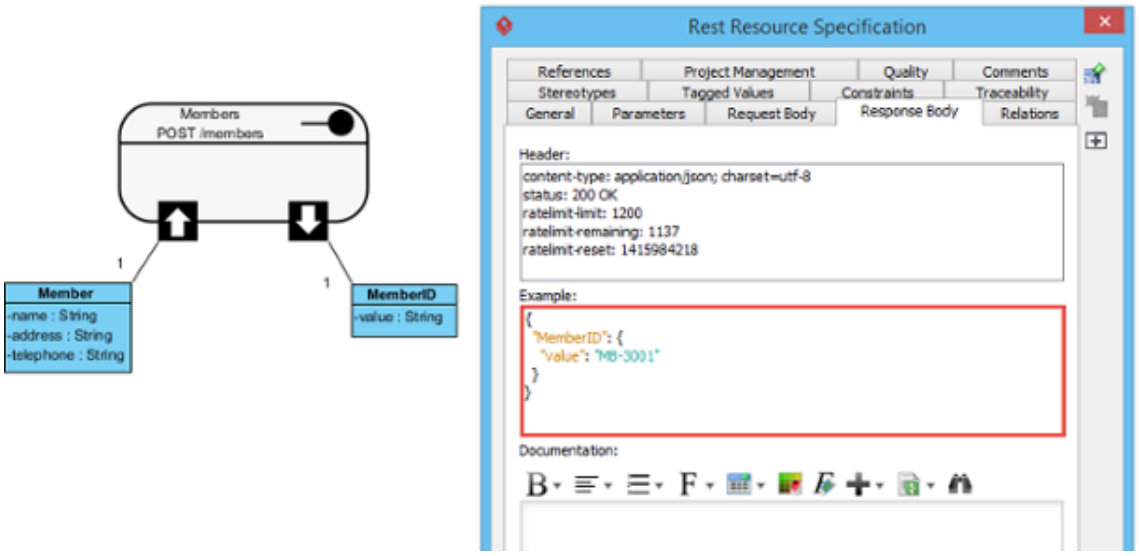
Ένα μέρος της μετατροπής του Postman JSON Collection σε OpenAPI, στηρίχθηκε στο πακέτο postman-to-openapi [15] της βιβλιοθήκης npm. Το πακέτο αυτό δέχεται ως όρισμα ένα JSON Postman Collection και το μετατρέπει σε αρχείο OpenAPI 3.0. Ακολουθούν συνοπτικά τα χαρακτηριστικά του:

- Βασικές Πληροφορίες του API: Για τη συμπλήρωση των πεδίων info [16] χρησιμοποιούνται οι πληροφορίες "name" και "description" του Collection. Για το version δεν υπάρχουν πληροφορίες και καθώς είναι υποχρεωτικό πεδίο, θα ελεγχθούν πρώτα τα collection variables σε περίπτωση που υπάρχει κάποια μεταβλητή με το όνομα version. Εάν υπάρχει θα χρησιμοποιηθεί η τιμή που βρέθηκε, διαφορετικά θα οριστεί η έκδοση 1.0.0.
- Folders - tags: Στο Postman υπάρχει η δυνατότητα ομαδοποίησης των requests σε φολδερς. Αν και αυτή η επιλογή δεν υπάρχει στο OpenAPI, χρησιμοποιούνται τα tags για αντίστοιχη ομαδοποίηση.
- Παράμετροι: Τα query και headers parameters μετατρέπονται σε OpenAPI Specification, με κύρια χαρακτηριστικά τα name, description και example για το περιεχόμενό τους. Το default schema είναι το string.
- Global Servers: Το πεδίο servers ορίζεται με τα urls που χρησιμοποιούνται στα requests.

- Responses: Μέσω του Postman είναι δυνατό να αποθηκευτούν responses ως examples. Τα responses αυτά θα περιέχονται στο πεδίο responses του request.

3.2 Επέκταση του πακέτου postman-to-openapi

Το πακέτο που αναφέρθηκε παραπάνω αποτέλεσε τη βάση όλης της μετατροπής. Για την περίπτωση που απαιτείται μόνο μία περιγραφή του Postman Collection σε OpenAPI μορφή, η χρήση μόνο του πακέτου είναι επαρκής.
Ωστόσο το αποτέλεσμα που προκύπτει δεν επαρκεί για την πλήρη απεικόνιση του REST API σε Class Diagram. Αυτό συμβαίνει διότι το body στις περιπτώσεις request και response δεν εμφανίζεται στο REST API του Visual Paradigm. Όπως αναφέρθηκε στην θεωρητική ανάλυση σχετικά με το Visual Paradigm το body πρέπει να εμφανίζεται και στο REST Specification ως example και ως κλάση συνδεδεμένη με αυτό, όπως φαίνεται παρακάτω:



Σχήμα 3.2: *Visual Paradigm Rest Response Body*

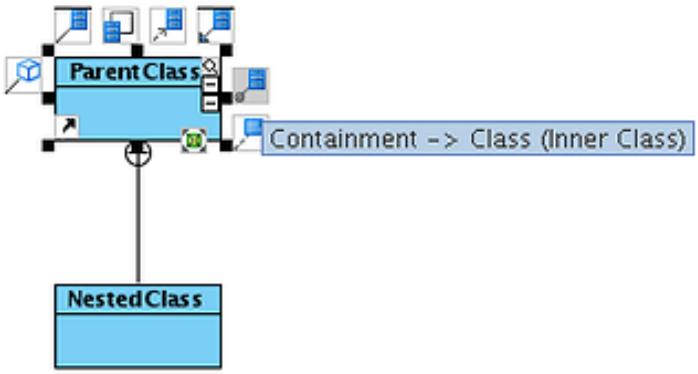
Η μετατροπή με τη χρήση του πακέτου postman-to-openapi δεν εμφανίζει καμία από τις δύο απεικονίσεις, με αποτέλεσμα το body να μην μεταφέρεται με κάποιο τρόπο στην REST απεικόνιση.

3.2.0.1 Προσθήκη Request και Response Body

Αρχικά, η απουσία του body από το example στο REST Specification και στις δύο περιπτώσεις οφείλεται στο πακέτο postman-to-openapi, καθώς η απεικόνιση αυτή αποτελεί κατά κάποιο τρόπο σύμβαση του Visual Paradigm. Επομένως, ο χρήστης θα πρέπει αφού εγκαταστήσει το πακέτο (npm install postman-to-openapi), να αντικαταστήσει το αρχείο index.js με ένα τροποποιημένο. Με αυτό τον τρόπο θα εμφανιστεί το body ως example.

Επίσης, όπως φαίνεται και στην εικόνα 3.2 το body πρέπει να απεικονίζεται και ως κλάση συνδεδεμένη στο REST API. Εάν δεν υπάρχει αυτή η σύνδεση, η πληροφορία του

body θα χαθεί σε περίπτωση εξαγωγής του REST API ως OpenAPI αρχείο. Σε αρχικό στάδιο προστέθηκε η σύνδεση του κάθε REST API που περιείχε request ή/και response body με μία κενή αντίστοιχη κλάση. Με αυτό τον τρόπο, κατά την εξαγωγή του REST API ως OpenAPI υπήρχε και η πληροφορία για το body, καθώς αυτό περιέχονταν στα examples. Η ύπαρξη των κλάσεων δήλωνε απλά την ύπαρξη του body.
Στη συνέχεια, προστέθηκαν και οι παράμετροι των bodies μέσα στις κλάσεις με το όνομα και τον τύπο τους. Ωστόσο, σε πολλές περιπτώσεις το body είναι σε nested μορφή [17]. Η απεικόνιση του nested τμήματος απεικονίζεται σε ένα διάγραμμα κλάσεων όπως στην εικόνα 3.3.



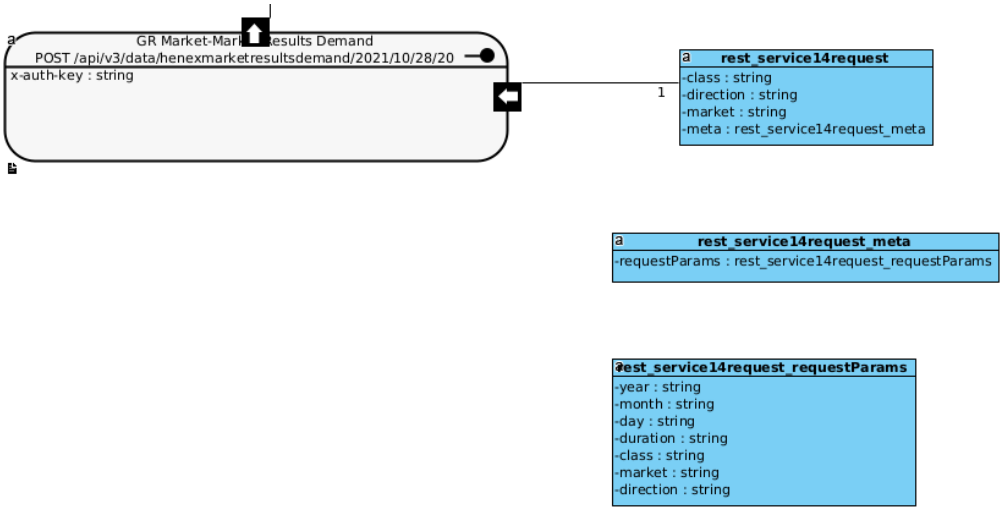
Σχήμα 3.3: *Nested JSON Body*

Αυτή η σύνδεση κατά το Reverse OpenAPI δεν είναι εμφανής, καθώς τα nested τμήματα εμφανίζονται ως ανεξάρτητες κλάσεις, παρόλο που στο documentation που εισάγεται (ως .yaml αρχείο) υπάρχουν οι αντίστοιχες αναφορές ανάμεσα στις κλάσεις.

Για παράδειγμα το ακόλουθο request body περιέχει τα πεδία meta και requestParams, τα οποία είναι εμφωλευμένα json τμήματα.

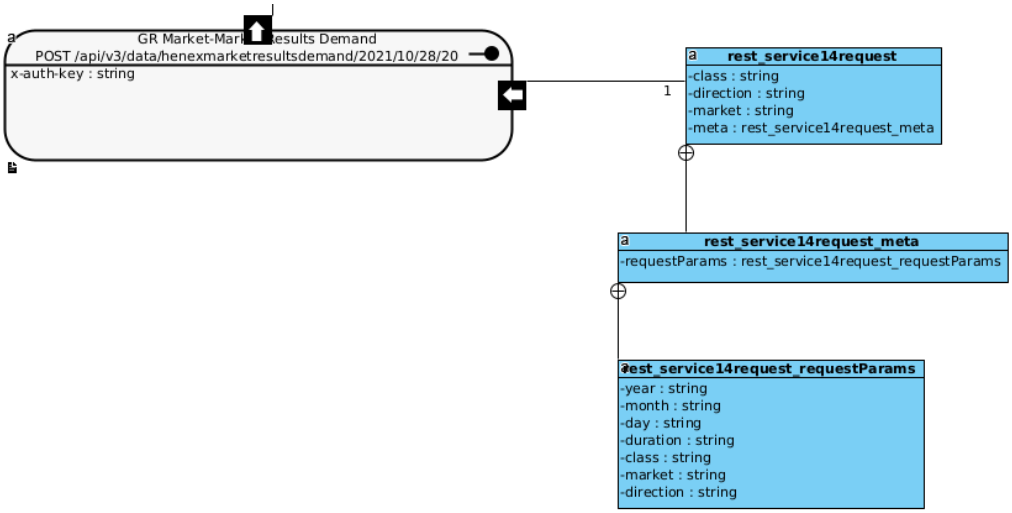
Σχήμα 3.4: *Nested JSON Body*

Τα τμήματα αυτά στο OpenAPI documentation που παράγεται από το σύστημα που δημιουργήθηκε δηλώνονται ως nested. Ωστόσο, η απεικόνιση τους στο Visual Paradigm είναι η εξής:



Σχήμα 3.5: *Nested Classes Visual Paradigm*

Παρατηρούμε δηλαδή, ότι συνδέεται μόνο το κύριο JSON κομμάτι και όχι τα εμφωλευμένα. Όπως θα αναλυθεί και παρακάτω στα βήματα που ακολουθούνται για τη χρήση του συστήματος, ο χρήστης θα πρέπει αφού ανεβάσει το OpenAPI αρχείο να συνδέσει ο ίδιος τα nested τμήματα των JSON Bodies, όπως φαίνεται παρακάτω:

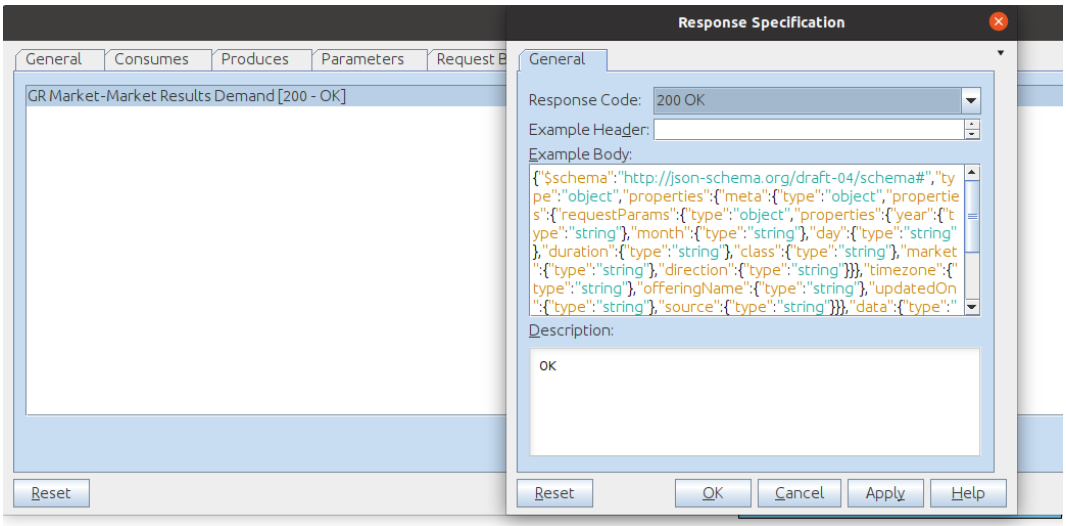


Σχήμα 3.6: *Nested Classes Visual Paradigm*

3.2.0.2 Αλλαγή Response Body

Το πακέτο postman-to-openapi που αναφέρθηκε παραπάνω, αναγνωρίζει το response body όπως και το request. Το response body στο JSON αρχείο του Postman Collection προέρχεται είτε από την αποθήκευση του response ως example, είτε από την εισαγωγή του από τον χρήστη ως example ξανά. Και στις δύο περιπτώσεις τα πεδία του body περιέχουν συγκεκριμένες τιμές ανάλογα με την κλίση και τις παραμέτρους του endpoint. Καθώς ο στόχος μας ήταν η παραγωγή documentation του API επιλέξαμε αντί για συγκεκριμένες τιμές, το response να έχει τη μορφή που είχε αρχικά αλλά να περιέχει τους τύπους των πεδίων του. Για να το πετύχουμε αυτό χρησιμοποιήθηκε το πακέτο generate-schema [18] της βιβλιοθήκης npm, το οποίο μετατρέπει JSON αντικείμενα σε JSON schemas. Η χρήση του προηγείται της μετατροπής του JSON Collection σε OpenAPI μορφή, καθώς ήταν πιο εύκολο να χρησιμοποιηθεί η JSON μορφή γι' αυτή τη μετατροπή. Το πακέτο αυτό φάνηκε ιδιαίτερα χρήσιμο, καθώς μετατρέπει ακόμα και nested json τμήματα σε json schemas, χωρίς να χρειάζονται άλλες αλλαγές από τον προγραμματιστή.

Ακολουθεί ένα παράδειγμα ενός example response στο Visual Paradigm, το οποίο έχει μετατραπεί σε JSON schema.

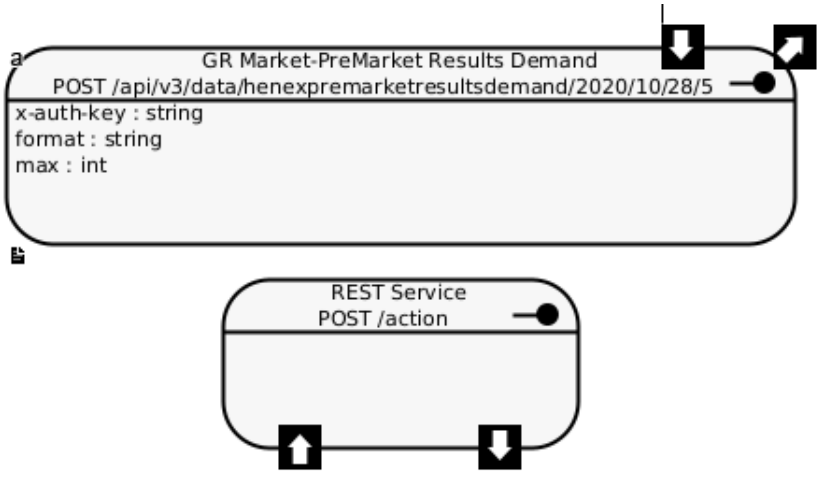


Σχήμα 3.7: *Visual Paradigm JSON Schema*

3.2.0.3 Λοιπές Αλλαγές

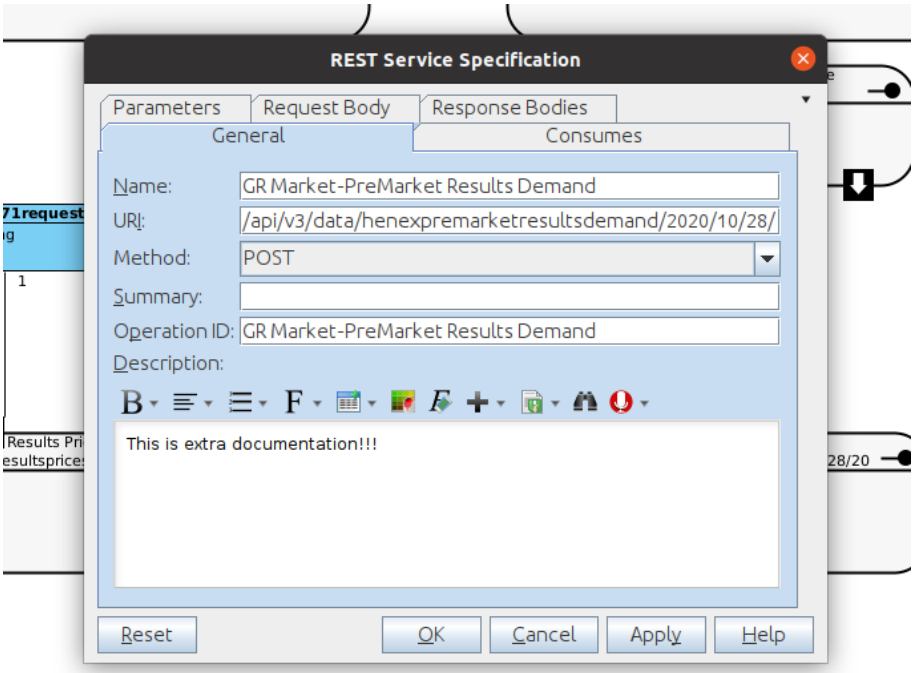
Πέρα των αλλαγών που περιγράφτηκαν παραπάνω, μέσω του συστήματός γίνονται και οι εξής αλλαγές:

- Προσθήκη ονόματος στο REST Service, μέσω του πεδίου operationId το οποίο συμπληρώνεται με τα στοιχεία του πεδίου summary.



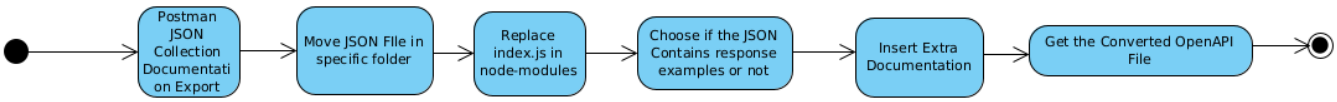
Σχήμα 3.8: *REST Service Name Before and After*

- Δυνατότητα στον χρήστη να προσθέτει επιπλέον documentation, το οποίο θα εμφανίζεται σε όλα τα REST Services, στο πεδίο Description.

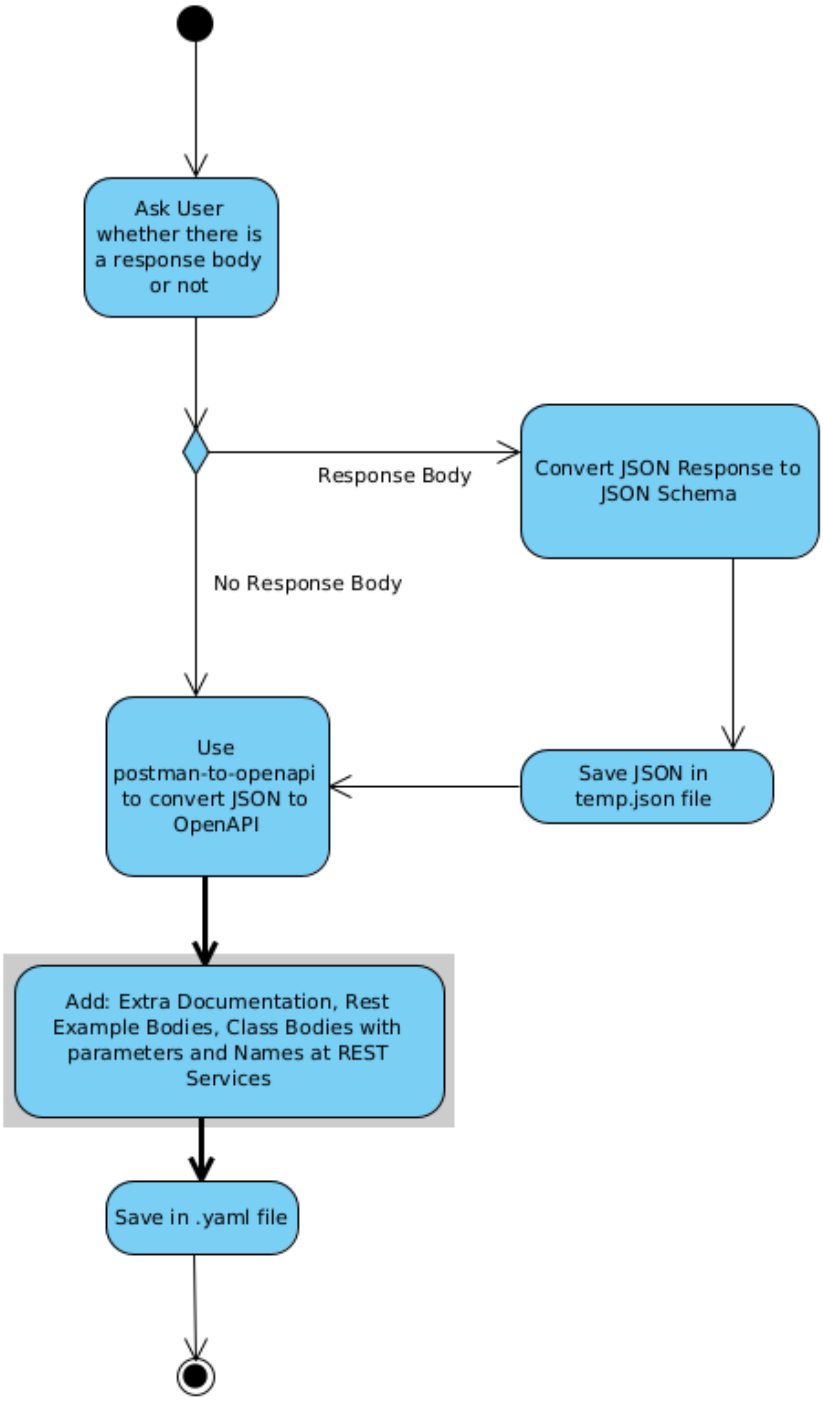


Σχήμα 3.9: *REST Service Extra Documentation*

Ακολουθεί η σχηματική αναπαράσταση της διαδικασίας χρήσης και λειτουργίας του συστήματος που αναπτύχθηκε μέσω ενός Activity Diagram. Η χρήση του θα παρουσιαστεί ξανά και στη συνέχεια μέσω ενός CLI και ενός απλού Web App.



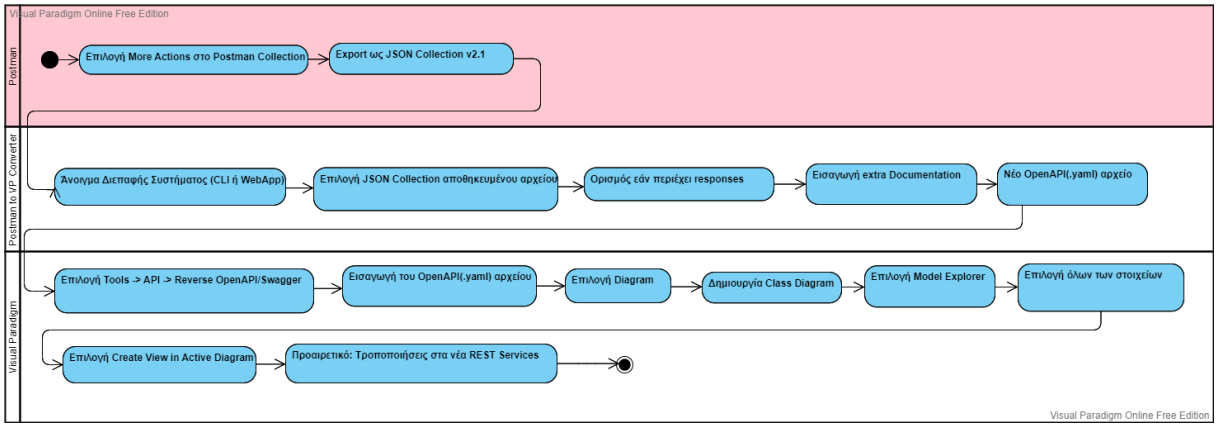
Σχήμα 3.10: *User System Usage Activity Diagram*



Σχήμα 3.11: *System Activity Diagram*

3.3 Παράδειγμα

Στο μέρος αυτό θα παρουσιαστεί ένα παράδειγμα μετατροπής ενός Postman Collection Documentation. Χρησιμοποιήθηκε ένα Collection από το Public API Network του Postman [19], το οποίο περιέχει απλά requests και bodies με χρήση μεθόδων GET, POST, PUT, DELETE.



Σχήμα 3.12: *Postman Actions*

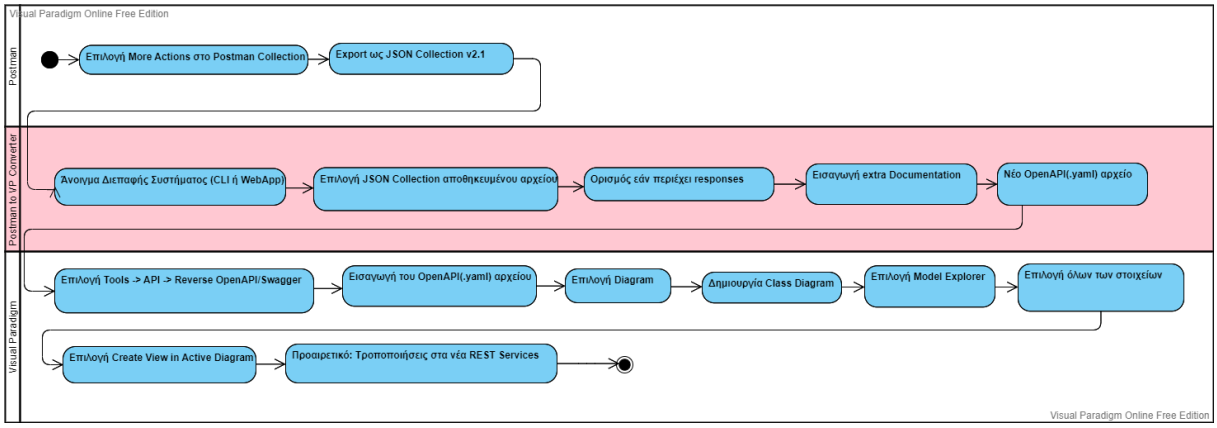
Ακολουθεί ένα κομμάτι του JSON Documentation.

```
{
	"info": {
		"_postman_id": "63a0159e-7bfb-46be-9e80-5e04d8a27723",
		"name": "Simple",
		"description": "Simple REST API developed with Lumen 5.8",
		"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
		"_exporter_id": "14129095"
	},
	"item": [
		{
			"name": "http://localhost:8000",
			"request": {
				"method": "GET",
				"header": [
					{
						"key": "x-api-key",
						"value": "OfJr4qriSrw7iZMpaWJm4OOrjGdfCgzrMQvTUkx7quaejgyokYy95yhnpjaLoiBT"
					}
				],
				"url": {
					"raw": "http://localhost:8000",
					"protocol": "http",
					"host": [
						"localhost"
					],
					"port": "8000"
				},
				"description": "REST API Index"
			},
			"response": []
		},
		{
			"name": "http://localhost:8000/v1/auth/register",
			"request": {
				"method": "POST",
				"header": [
					{
						"key": "Content-Type",
						"value": "application/json"
					},
					{
						"key": "x-api-key",
						"value": "OfJr4qriSrw7iZMpaWJm4OOrjGdfCgzrMQvTUkx7quaejgyokYy95yhnpjaLoiBT"
					}
				],
				"body": {
					"mode": "raw",
					"raw": "{\n\t\"first_name\": \"Fru\",\n\t\"last_name\": \"Kerick\",\n\t\"email\":
\"frukerickjeff@gmail.com\",\n\t\"password\": \"secret23\",\n\t\"password_confirmation\": \"secret23\"\n}"
```


Σχήμα 3.13: *JSON Collection*

Το συγκεκριμένο Collection δεν περιέχει response examples.

Μετά την επεξεργασία του μέσω του συστήματός προκύπτει ένα OpenAPI (.yaml) αρχείο.



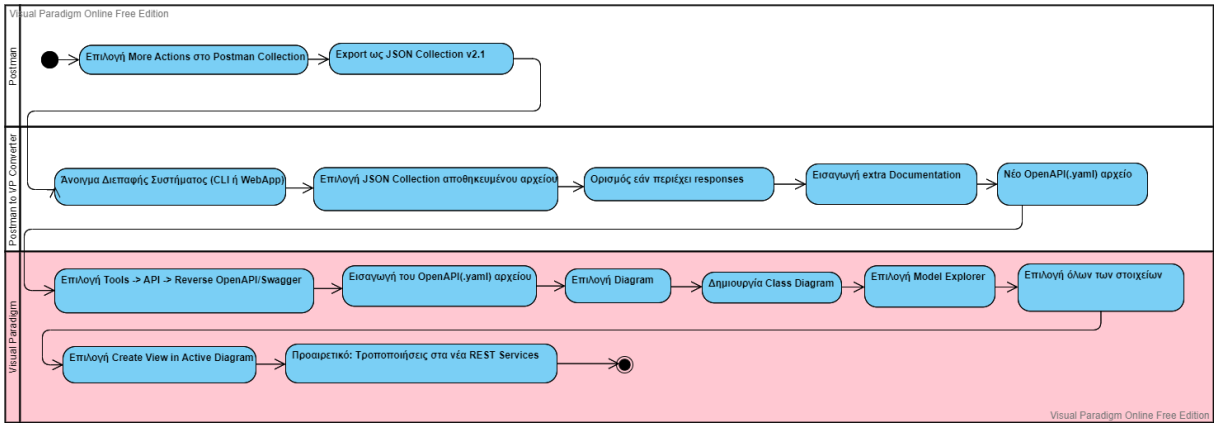
Σχήμα 3.14: *Converter Actions*

Ακολουθεί ένα τμήμα του αρχείου:

```yaml
openapi: 3.0.0
info:
  title: Simple
  description: Simple REST API developed with Lumen 5.8
  version: 1.0.0
servers:
  - url: http://localhost:8000
paths:
  /:
    get:
      tags:
        - General
      summary: http://localhost:8000
      operationId:  http://localhost:8000
      description:
      description: REST API Index
      parameters:
        - name: x-api-key
          in: header
          schema:
            type: string
          example: OfJr4qriSrw7iZMpaWJm4OOrjGdfCgzrMQvTUkx7quaejgyokYy95yhnpjaLoiBT
      responses:
        '200':
          description: Successful response
          content:
  /v1/auth/register:
    post:
      tags:
        - General
      summary: http://localhost:8000/v1/auth/register
      operationId:  http://localhost:8000/v1/auth/register
      description:
      description: Register a new user in the database
      x-codegen-request-body-name: rest_service32request
      requestBody:
        content:
          application/json:
            schema:
              type: object
              $ref: "#/components/schemas/rest_service32request"
            example:
              first_name: Fru
              last_name: Kerick
              email: frukerickjeff@gmail.com
              password: secret23
              password_confirmation: secret23
      parameters:
        - name: Content-Type
          in: header
```

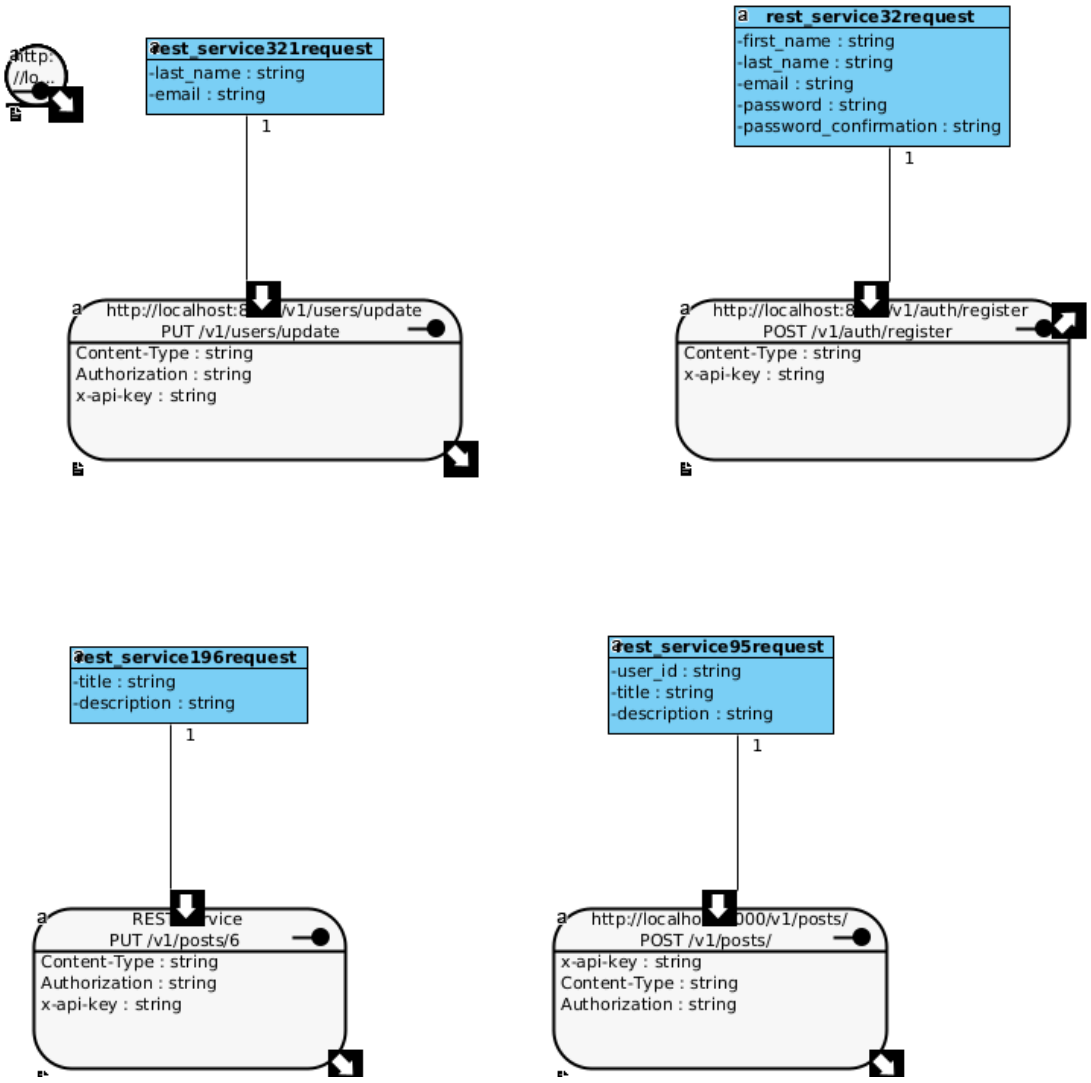
Σχήμα 3.15: *OpenAPI Documentation*

Τέλος, ακολουθεί η αναπαράστασή του ως REST API, μέσω του Reverse OpenAPI tool στο Visual Paradigm.

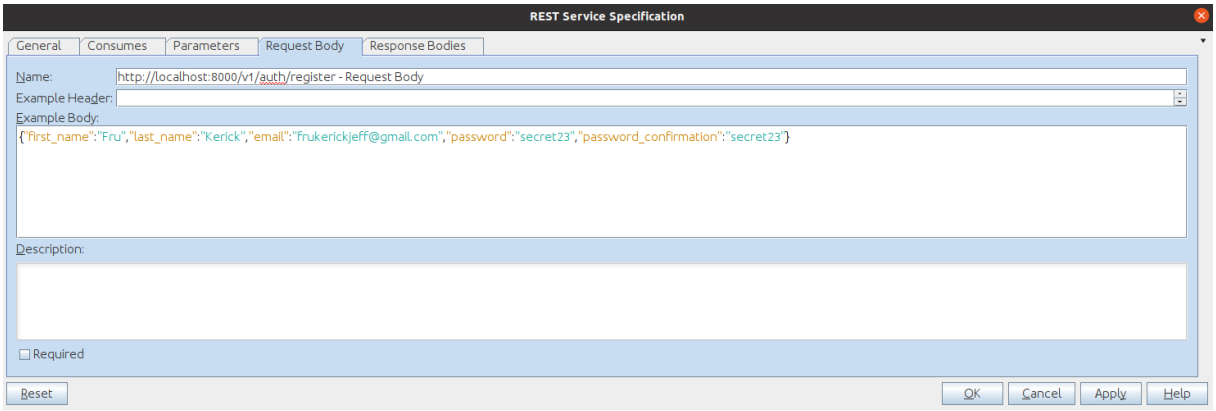


Σχήμα 3.16: *Visual Paradigm Actions*

Όπως φαίνεται και στην εικόνα, έχουν δημιουργηθεί τα REST Services που περιέχουν τα ονόματα των παραμέτρων και τους τύπους τους. Ακόμα, φαίνονται οι κλάσεις των request bodies με τα ονόματα των πεδίων και τους τύπους τους.



Σχήμα 3.17: *REST API Visual Paradigm*



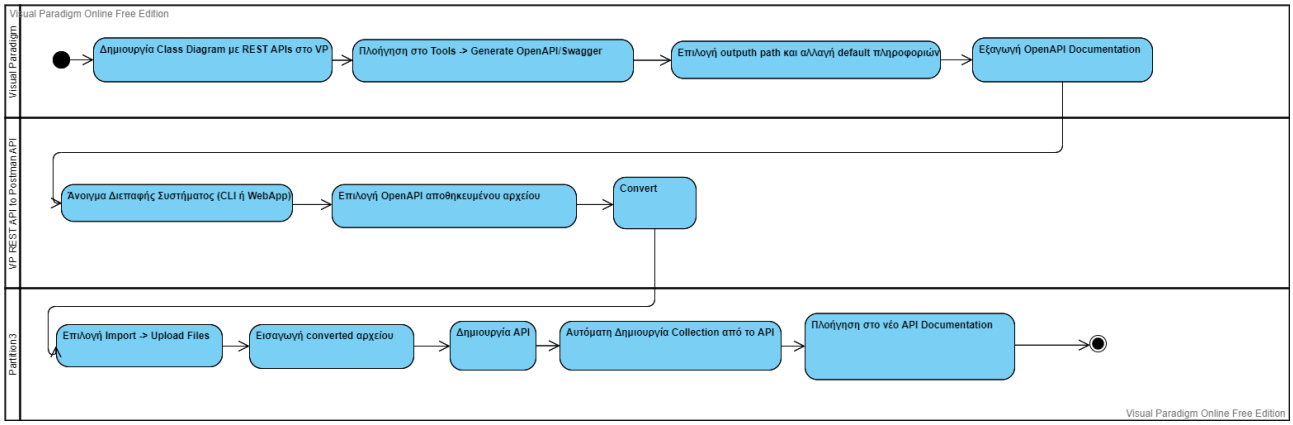
Σχήμα 3.18: *REST API Visual Paradigm Example Body*

Κεφάλαιο 4

Visual Paradigm REST API to Postman API

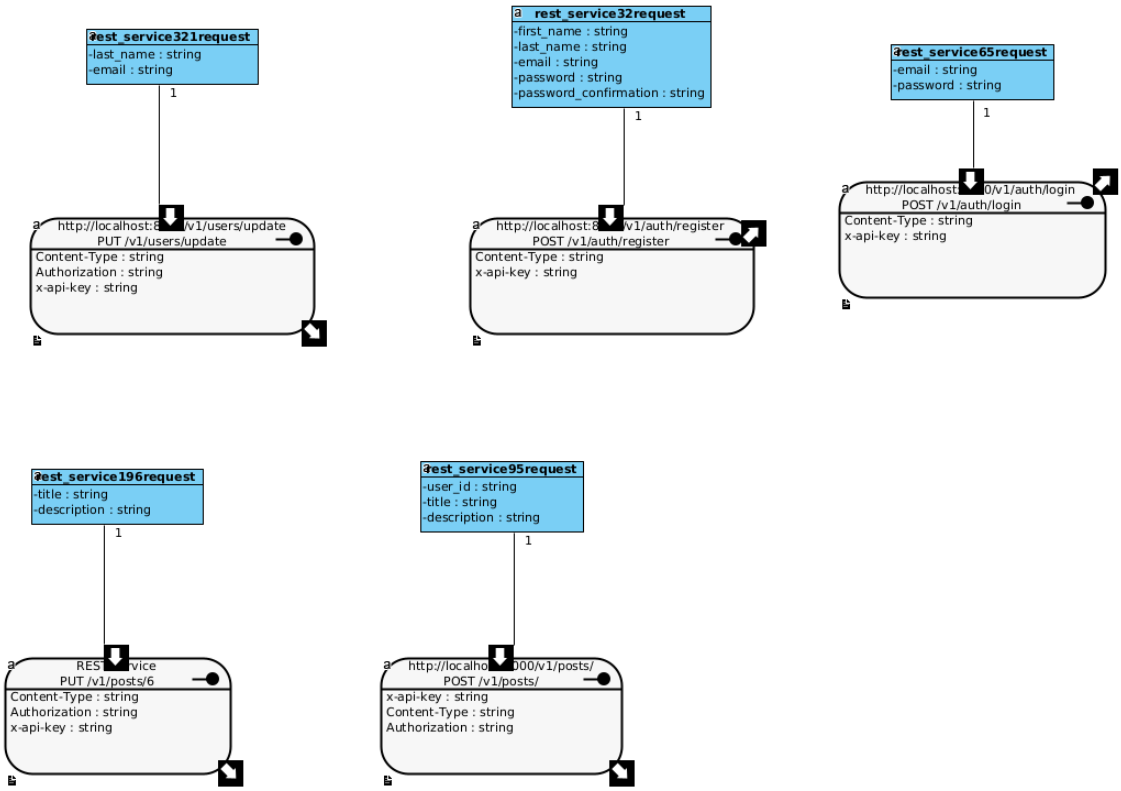
Στο κεφάλαιο αυτό περιγράφεται η υλοποίηση του συστήματος, μέσω του οποίου από ένα REST API σχεδιασμένο στο Visual Paradigm δημιουργούμε ένα Postman API, και κατά συνέπεια ένα Postman Collection και το αντίστοιχο Documentation του.
Η διαδικασία είναι αρκτετά πιο σύντομη σε σχέση με αυτή της προηγούμενης ενότητας καθώς η μορφή των αρχείων που θα χρησιμοποιηθούν βρίσκονται ήδη σε OpenAPI μορφή.

Ακολουθεί η σχηματική περιγραφή μέσω ενός Activity Diagram των βημάτων που ακολουθούν:



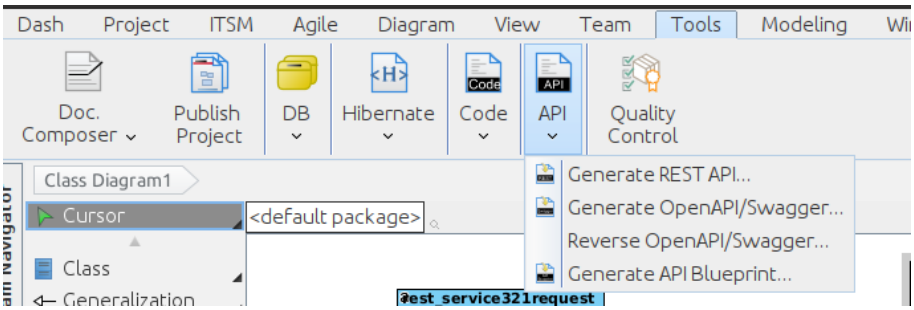
Σχήμα 4.1: *VP REST API to Postman API*

Επομένως, έστω ότι έχουμε σχεδιάσει στο Visual Paradigm ένα REST API όπως το ακόλουθο:



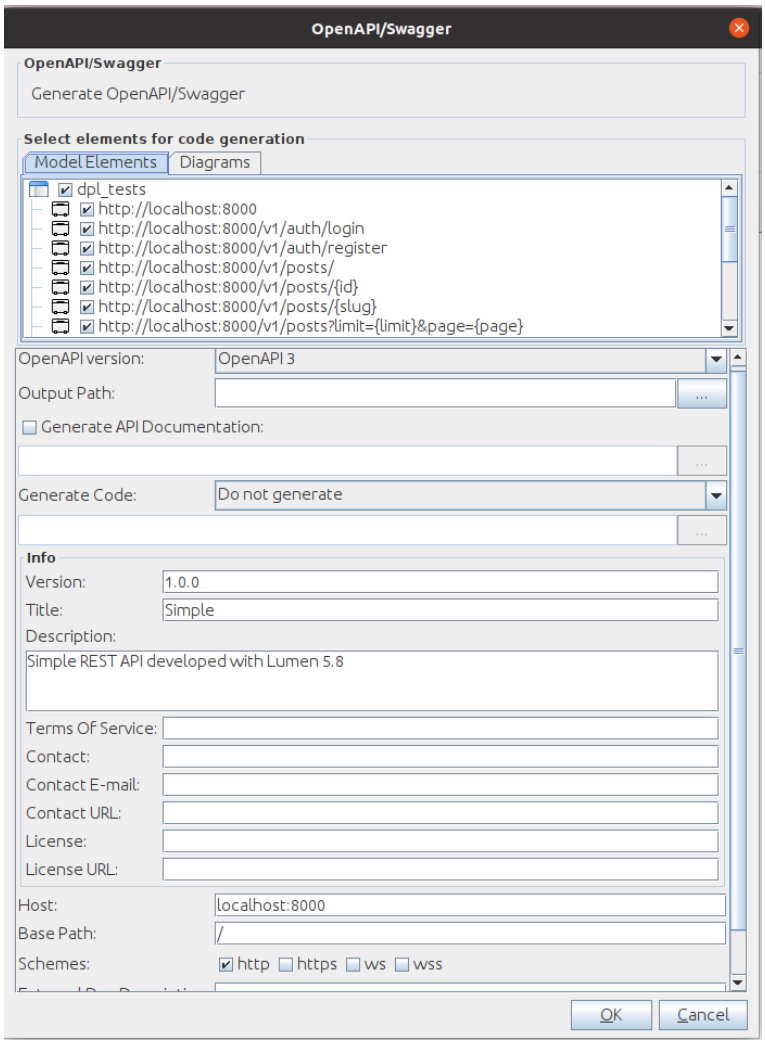
Σχήμα 4.2: *REST API Visual Paradigm*

Πλοηγούμαστε στο Tools → *GenerateOpenAPI/Swagger*.

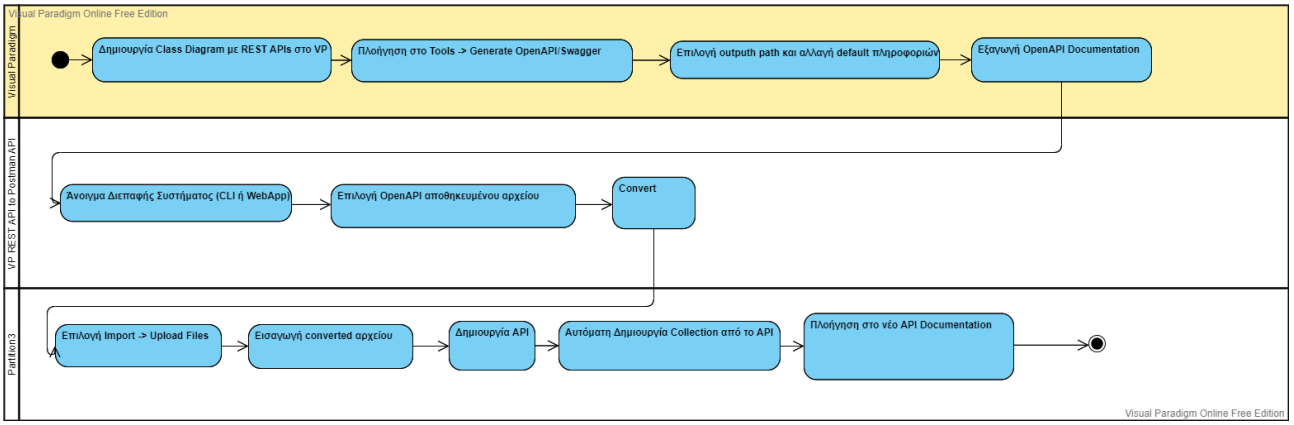


Σχήμα 4.3: *REST API Visual Paradigm*

Μας εμφανίζεται το παράθυρο της εικόνας 4.9. Τα Model Elements είναι τα REST APIs που έχουν δημιουργηθεί. Επιλέγουμε ένα .yaml αρχείο από το Output Path, αλλάζουμε τις default πληροφορίες, όπως τα version, title, Host και πατάμε OK. Το αποτέλεσμα είναι το Documentation σε μορφή OpenAPI.

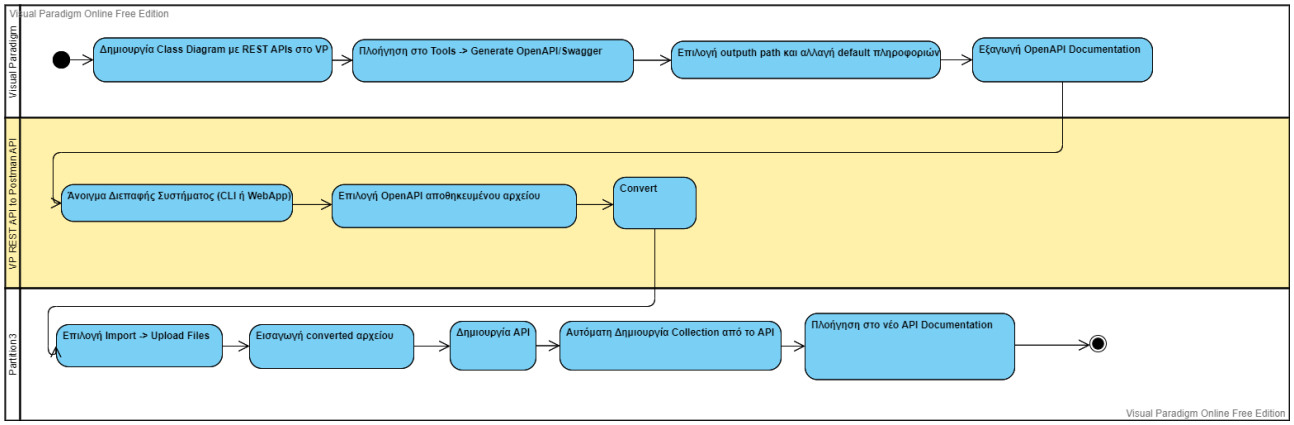


Σχήμα 4.4: *REST API Visual Paradigm Generate OpenAPI*



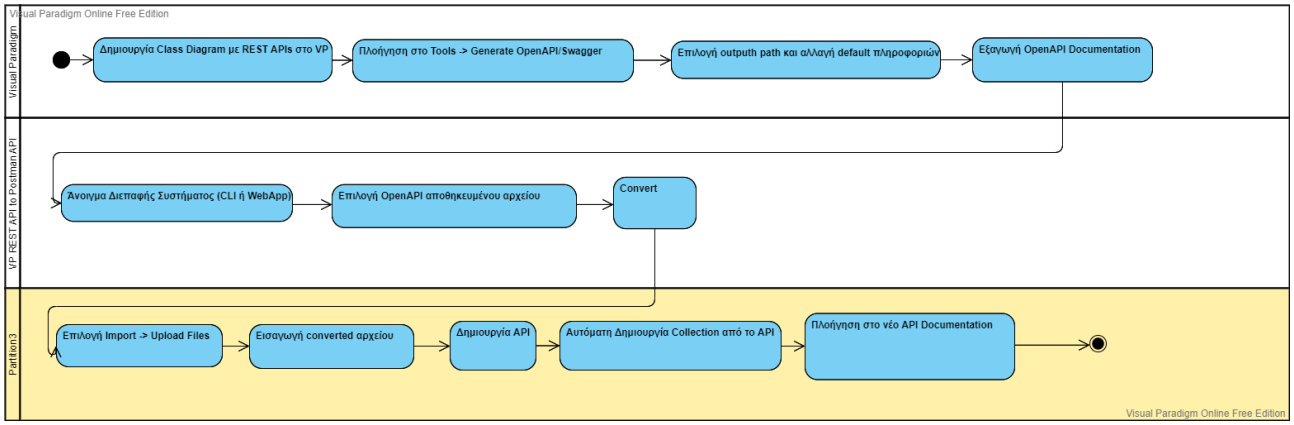
Σχήμα 4.5: *VP REST API to Postman API - Visual Paradigm*

Η μορφή του Documentation αμέσως μετά το Visual Paradigm δεν είναι πλήρως σωστή ώστε να ανέβει στο Postman. Γι' αυτό το λόγο, επεξεργάζεται μέσω του αρχείου yaml_to_postman.js. Οι αλλαγές αφορούν κυρίως τη μορφή των bodies των requests και responses.



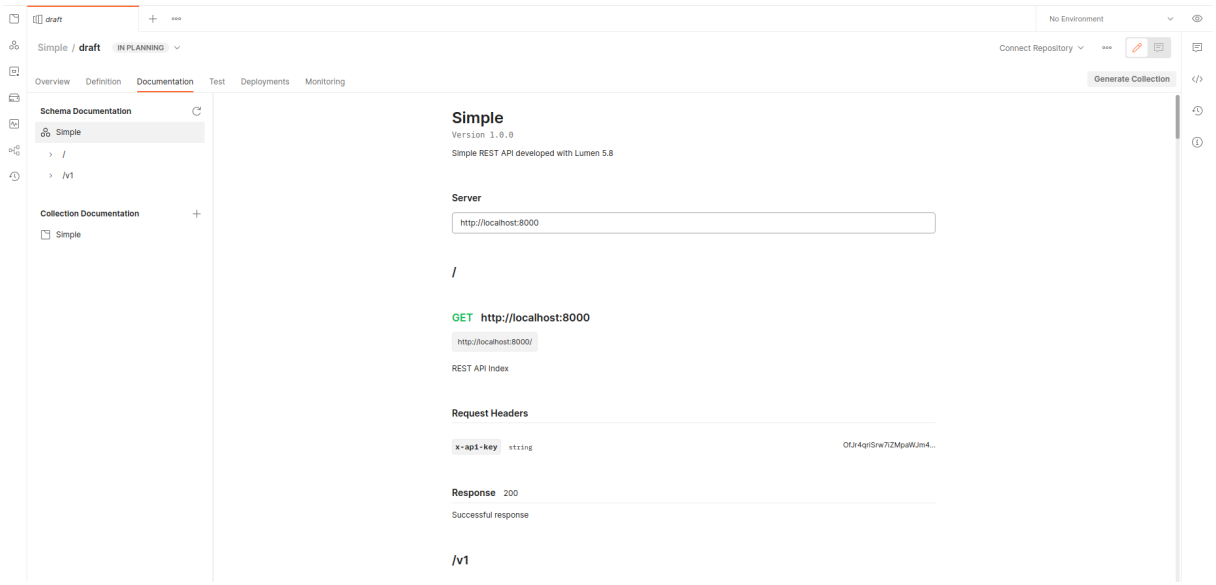
Σχήμα 4.6: *VP REST API to Postman API - Converter*

Ύστερα, το αρχείο που προκύπτει μπορεί να ανέβει μέσω του Import στο Postman, από αυτό αυτόματα να δημιουργηθεί ένα API και από το API ένα Collection με όλες τις πληροφορίες που είχαν οριστεί εξ' αρχής μέσα στο REST API.

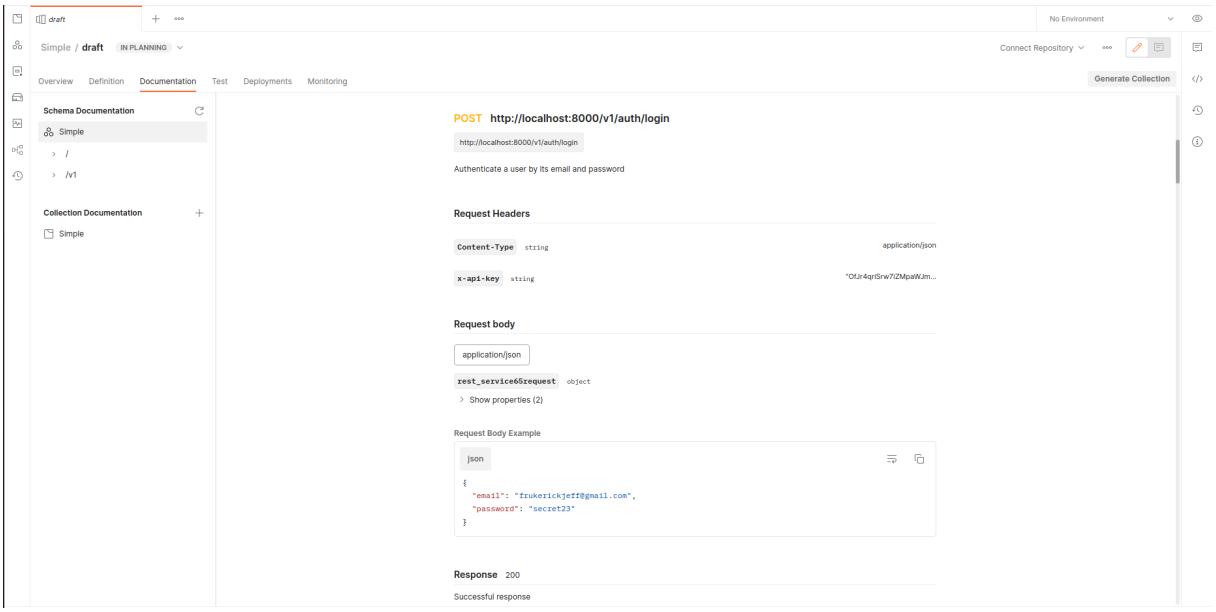


Σχήμα 4.7: *VP REST API to Postman API - Postman*

Το πιο σημαντικό από αυτή τη διαδικασία, είναι το γεγονός ότι μέσω της δημιουργίας του API στο Postman προκύπτει και το Documentation του, όπως φαίνεται παρακάτω:



Σχήμα 4.8: *Postman API Documentation*



Σχήμα 4.9: *Postman API Documentation*

Κεφάλαιο 5

Χρήση Συστήματος μέσω CLI

Στο κεφάλαιο αυτό παρουσιάζεται ο τρόπος χρήσης του συστήματος που περιγράφτηκε παραπάνω μέσω ενός απλού CLI (Command Line Interface). Το CLI αναπτύχθηκε σε γλώσσα Javascript. Για τα γραφικά χρησιμοποιήθηκαν τα πακέτα chalk [20] και figlet [21] της βιβλιοθήκης npm.

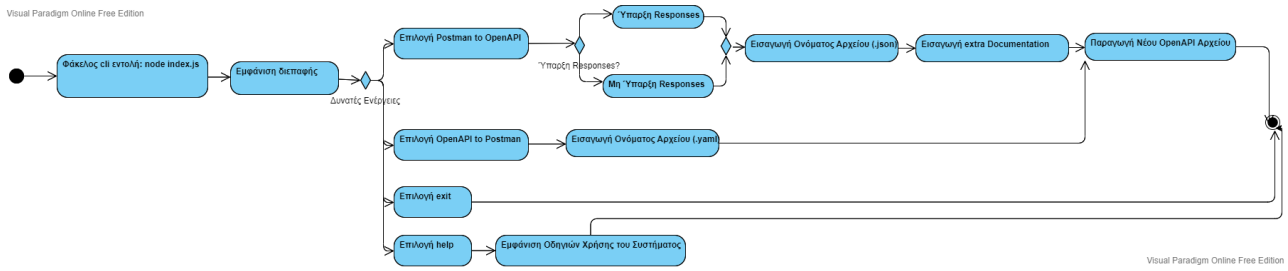
5.1 Πρόσβαση στο Project

Το project μπορεί να βρεθεί και να χρησιμοποιηθεί μέσω του GitHub. Ακολουθούν τα βήματα προεργασίας για τη χρήση του:

Σχήμα 5.1: *CLI Install*

1. Μέσω ενός terminal δίνεται η εντολή git clone https://github.com/nafsika24/diploma_-thesis.git
2. Μέσω ενός προγράμματος επεξεργασίας, όπως το Visual Studio Code [22], ανοίγεται το project.
3. Μέσω του terminal στο αρχικό directory (diploma_thesis) δίνεται η εντολή npm install, ώστε να εγκατασταθούν όλα τα πακέτα που απαιτούνται.
4. Μέσα στον φάκελο node_modules/postman_to_openapi/lib αντικαθίσταται το αρχείο index.js με το αντίστοιχο αρχείο που βρίσκεται στον φάκελο postman_to_openapi.
5. Μέσω του terminal στο directory diploma_thesis/cli δίνεται η εντολή npm install, ώστε να εγκατασταθούν όλα τα πακέτα που απαιτούνται για το CLI.

5.2 Χρήση CLI



Σχήμα 5.2: *CLI Usage*

Αφού πραγματοποιηθεί η παραπάνω προεργασία το σύστημα είναι έτοιμο για χρήση. Μέσα από τον φάκελο cli δίνεται η εντολή node index.js.

Σχήμα 5.3: *CLI Main*

Ο χρήστης έχει τις εξής επιλογές:

1. Postman to OpenAPI: Μετατροπή ενός αρχείου Postman JSON Collection Documentation σε OpenAPI με σκοπό την απεικόνιση του ως REST API στο Visual Paradigm.
2. OpenAPI to Postman: Μετατροπή ενός αρχείου OpenAPI το οποίο έχει παραχθεί από ένα REST API του Visual Paradigm σε OpenAPI αρχείο κατάλληλο για δημιουργία ενός API στο Postman.
3. exit: Έξοδος από το σύστημα.
4. help: Οδηγίες χρήσης του συστήματος.

Θα αναλυθούν οι δύο πρώτες επιλογές:

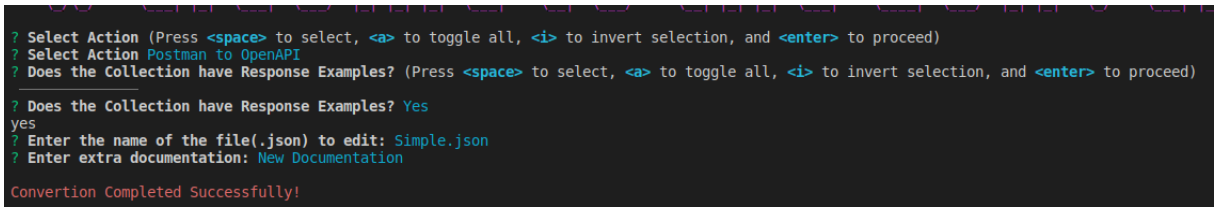
5.2.1 Postman to OpenAPI

Αφού ο χρήστης επιλέξει αυτή την ενέργεια, θα του εμφανιστεί η επιλογή σχετικά με το εάν ο αρχείο που θα μετατραπεί περιέχει responses:

Ανεξάρτητα από το ποια επιλογή θα διαλέξει, στη συνέχεια θα εμφανιστεί η ερώτηση σχετικά με το όνομα του αρχείου που θα μετατραπεί και εάν επιθυμεί να προσθέσει κάποιο ακόμα Documentation στα REST APIs. Εάν η μετατροπή είναι επιτυχής εμφανίζεται το αντίστοιχο μήνυμα όπως στην εικόνα 5.5. Το αρχείο που θα δωθεί ως είσοδος πρέπει να

Σχήμα 5.4: *CLI Postman to OpenAPI Responses*

βρίσκεται στον φάκελο files μέσα στο cli. Σε περίπτωση που δεν υπάρχει εμφανίζεται μήνυμα λάθους. Το αρχείου παράγεται θα βρίσκεται επίσης στον φάκελο files με το ίδιο όνομα και την κατάληξη forVisual.yaml.



Σχήμα 5.5: *CLI Postman to OpenAPI*

5.2.2 OpenAPI to Postman

Αφού ο χρήστης επιλέξει αυτή την ενέργεια, θα εμφανιστεί η ερώτηση σχετικά με ο όνομα του αρχείου που θα μετατραπεί, το οποίο θα πρέπει να βρίσκεται στον φάκελο files. Το τελικό αρχείο έχει το ίδιο όνομα με το αρχικό με προσθήκη της κατάληξης forPostman.

Σχήμα 5.6: *CLI OpenAPI to Postman*

Κεφάλαιο 6

Χρήση Συστήματος μέσω Web App

Εκτός από τη χρήση του συστήματος μέσω CLI, αναπτύχθηκε και ένα απλό Web Application με χρήση του framework ReactJS [23].

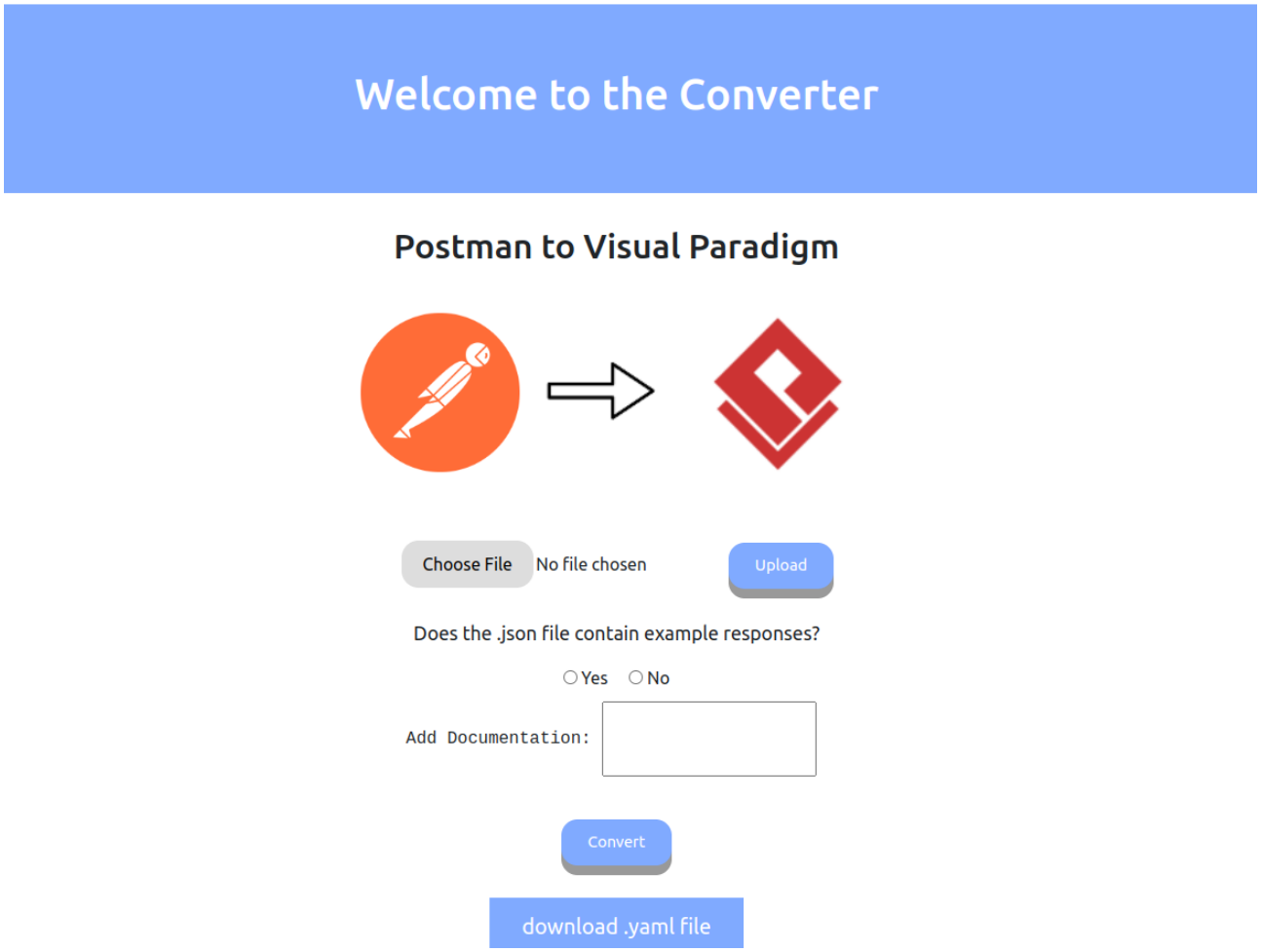
6.1 Πρόσβαση στο Project

Το project μπορεί να βρεθεί και να χρησιμοποιηθεί μέσω του GitHub. Ακολουθούν τα βήματα προεργασίας για τη χρήση του:

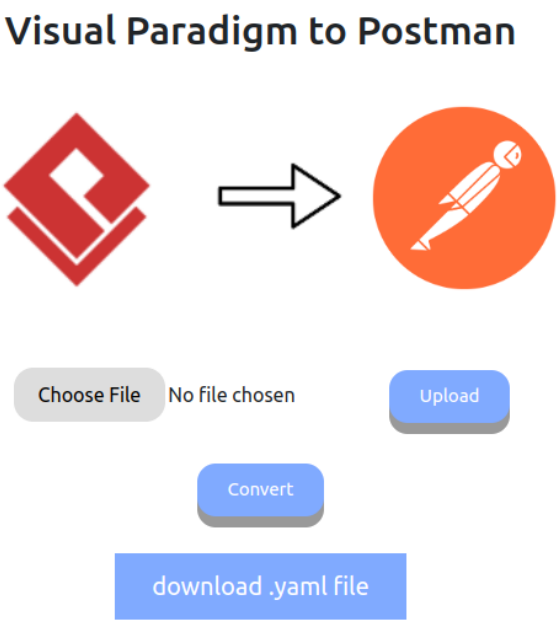
1. Μέσω ενός terminal δίνεται η εντολή git clone https://github.com/nafsika24/diploma_thesis_webapp.git
2. Μέσω ενός προγράμματος επεξεργασίας, όπως το Visual Studio Code [22], ανοίγεται το project.
3. Μέσω του terminal στα directories (api, client) δίνεται η εντολή npm install, ώστε να εγκατασταθούν όλα τα πακέτα που απαιτούνται.
4. Μέσα στον φάκελο node_modules/postman_to_openapi/lib του directory api αντικαθίσταται το αρχείο index.js με το αντίστοιχο αρχείο που βρίσκεται στον φάκελο postman_to_openapi.

6.2 Χρήση Web App

Για να χρησιμοποιηθεί το Web App, πρέπει να ανοίξουν δύο terminals, ένα μέσα στο φάκελο api και ένα μέσα στο φάκελο client και να δωθεί η εντολή npm start. Ύστερα, ανοίγει το Web App, όπως φαίνεται παρακάτω:



Σχήμα 6.1: *Web App*



Σχήμα 6.2: *Web App*

Υπάρχουν δύο δυνατές ενέργειες, η μετατροπή ενός Postman JSON Collection αρχείου σε OpenAPI με στόχο την απεικόνισή του στο Visual Paradigm και η μετατροπή ενός OpenAPI αρχείου από το Visual Paradigm σε OpenAPI με στόχο την δημιουργία ενός API στο Postman. Η ανάλυση σχετικά με τον λόγο ύπαρξης ορισμένων επιλογών αναλύεται παραπάνω. Επομένως, εδώ θα παρουσιαστούν συνοπτικά οι ενέργειες που μπορεί ο χρήστης να εκτελέσει.

6.2.1 Postman JSON Collection to Visual Paradigm

Όπως φαίνεται και στην εικόνα 6.1 ο χρήστης επιλέγει ένα αρχείο και μέσω του Upload το ανεβάζει στο σύστημα. Ύστερα επιλέγει εάν το αρχείο περιέχει response examples και εάν θα προστεθεί κάποια νέα τεκμηρίωση και πατάει Convert.
Τέλος, μπορεί να κατεβάσει το νέο αρχείο μέσω του κουμπιού download .yaml file.

6.2.2 REST API Visual Paradigm to Postman API

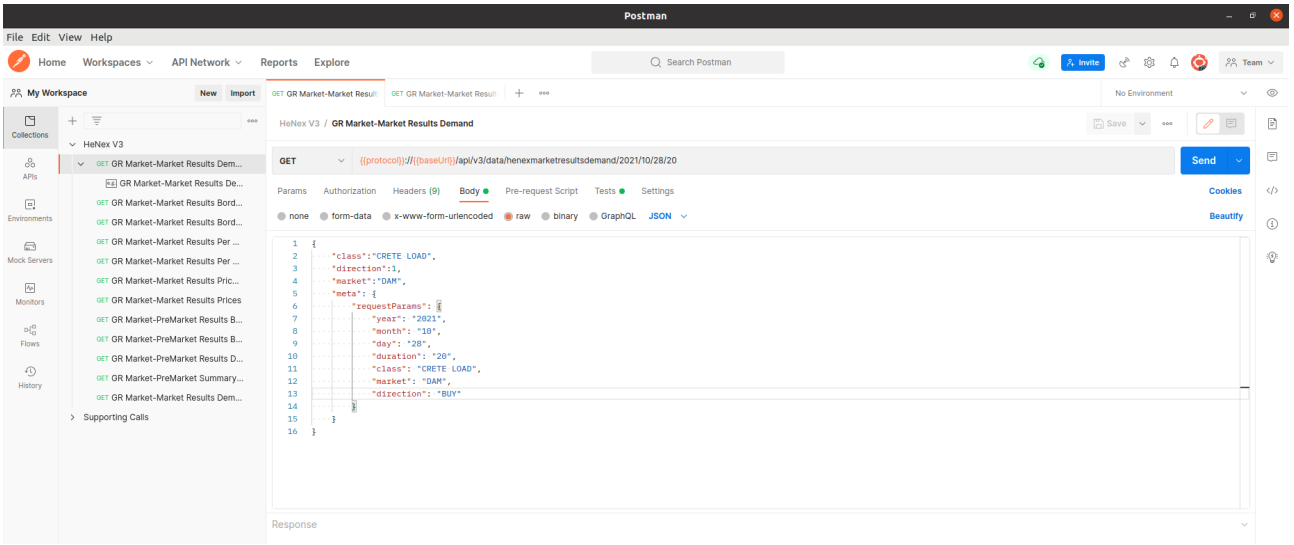
Όπως φαίνεται και στην εικόνα 6.2 ο χρήστης επιλέγει ένα αρχείο και μέσω του Upload το ανεβάζει στο σύστημα και πατάει Convert για να μετατραπεί. Τέλος, μπορεί να κατεβάσει το νέο αρχείο μέσω του κουμπιού download .yaml file.

Κεφάλαιο 7

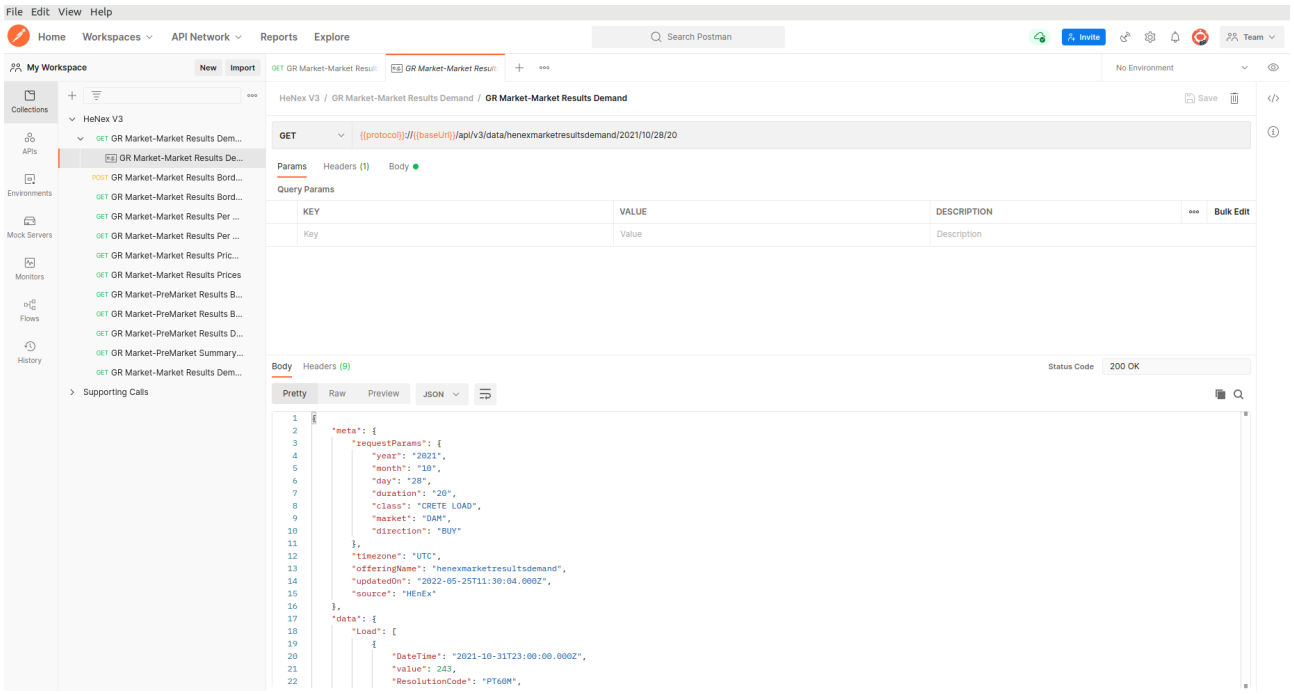
Αποτελέσματα

Στο καφάλαιο αυτό θα παρουσιάσουμε όλες τις διαδικασίες και τα αποτελέσματά τους που περιγράφτηκαν στις προηγούμενες ενότητες. Αρχικά, θα ξεκινήσουμε από ένα Postman Collection, θα το μετατρέψουμε σε OpenAPI Documentation και θα το αναπαραστήσουμε στο Visual Paradigm. Στη συνέχεια θα δείξουμε την αντίστροφη διαδικασία, δηλαδή πως από την αναπαράσταση στο Visual Paradigm δημιουργούμε ένα API και οπτικοποιούμε το Documentation του.

Θα χρησιμοποιήσουμε ένα ολοκληρωμένο Postman Collection, το οποίο αν και περιέχει GET μεθόδους εξόρισμού, επειδή περιέχει request bodies αυτές στη συνέχεια θα ερμηνευτούν ως POST από το Visual Paradigm. Το Collection αυτό αποτελεί μία συλλογή από requests με path και query parameters, μεταβλητές, καθώς και nested request και response bodies. Ακολουθεί η παρουσίασή του στο Postman:



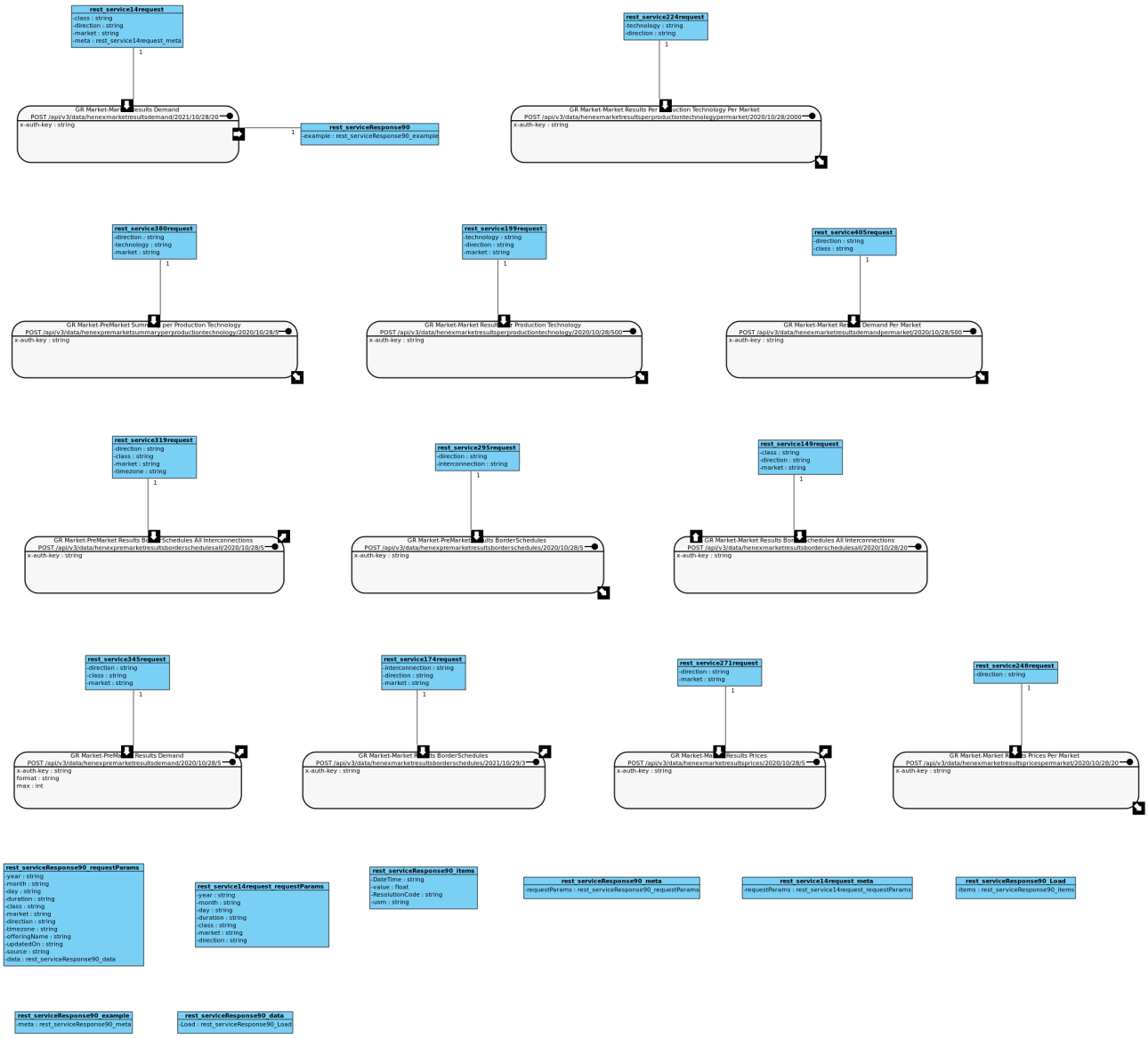
Σχήμα 7.1: *Postman Collection*



Σχήμα 7.2: *Postman Collection Response*

Στη συνέχεια κάνουμε export το Collection και έχουμε το JSON Documentation του.

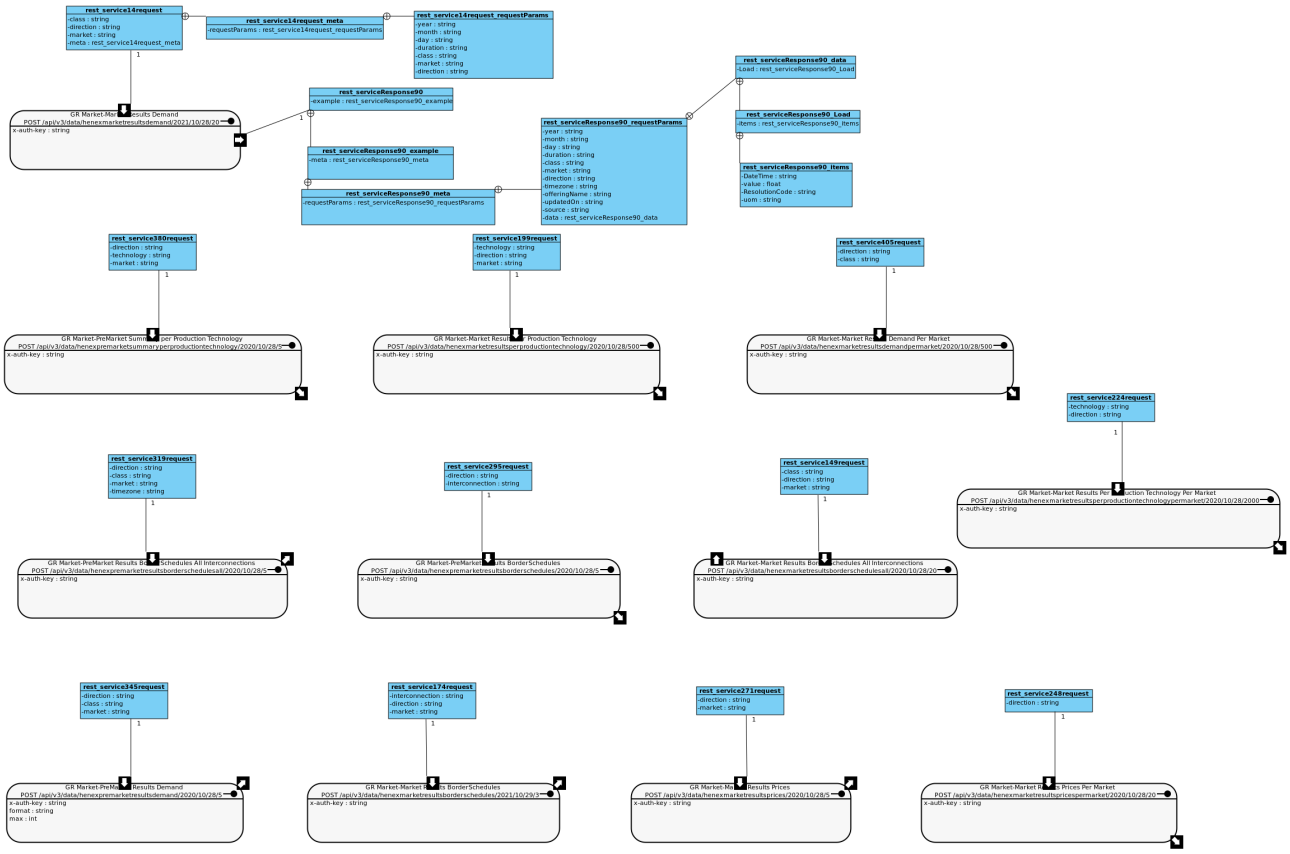
Με τη χρήση του συστήματός που δημιουργήσαμε μέσω του CLI ή του WebApp, παίρνουμε το API OpenAPI Documentation, το οποίο το κάνουμε import στο Visual Paradigm. Ακολουθεί η αναπαράστασή του:



Σχήμα 7.3: *VP Presentation*

Όπως βλέπουμε και στην αναπαράσταση στο Visual Paradigm το αρχικό Postman Collection έχει μετατραπεί πλήρως σε API και αναπαρασταθεί με όλα τα στοιχεία του. Παρατηρούμε ότι υπάρχουν κάποια components στο τέλος που δεν συνδέονται μεταξύ τους. Το γεγονός αυτό έχει σχολιαστεί και πιο πάνω και πρόκειται για την περίπτωση που έχουμε body (request, response) σε nested μορφή. Η πληροφορία υπάρχει για τις nested περιπτώσεις και φαίνεται στο specification του REST API.

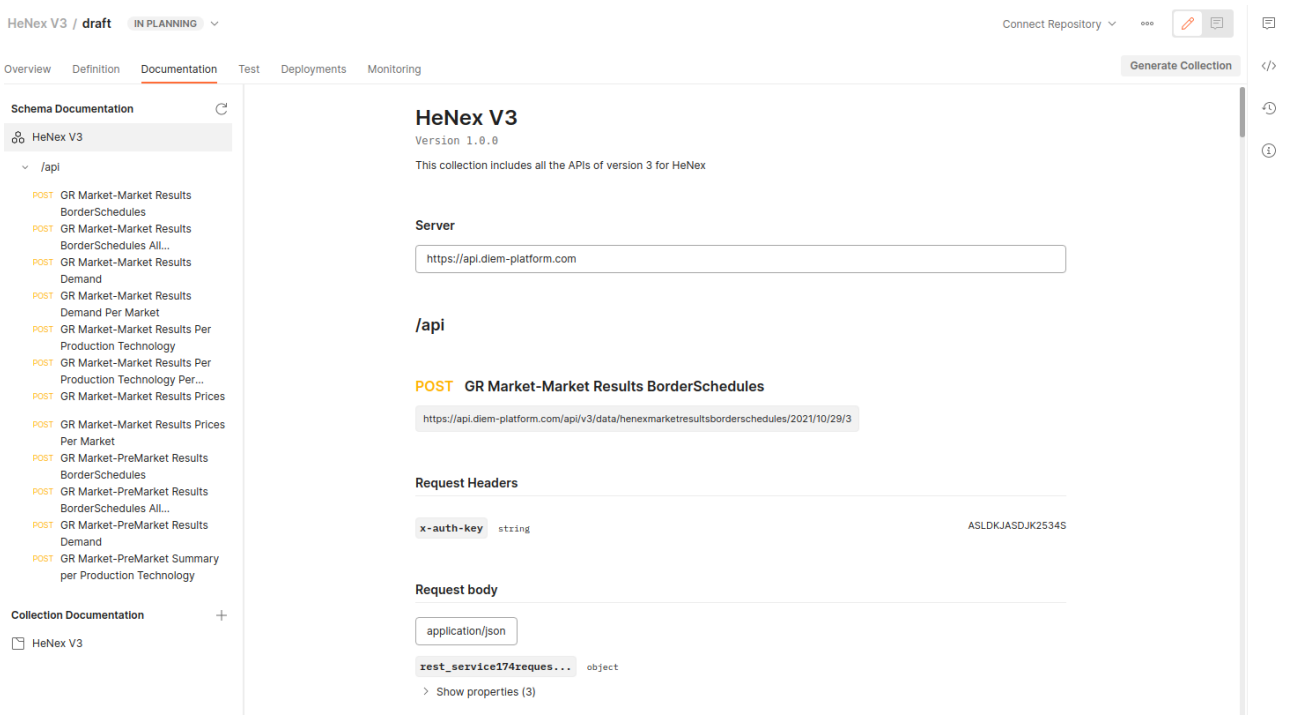
Ο χρήστης χειροκίνητα μπορεί να ενώσει τις κλάσει όπως φαίνεται και παρακάτω:



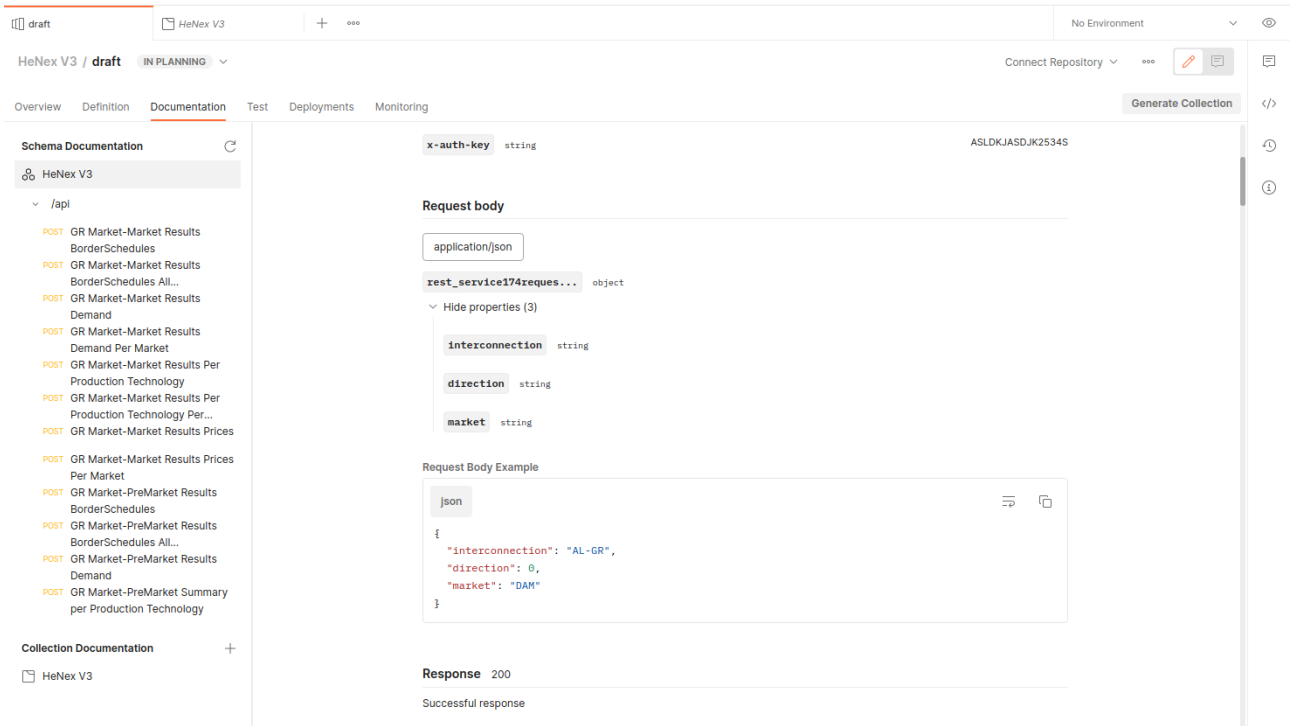
Σχήμα 7.4: *VP Presentation Nested Classes*

Στη συνέχεια, κάνουμε export το OpenAPI και προκύπτει ένα .yaml αρχείο με το Documentation του API. Το αρχείο αυτό το επεξεργαζόμαστε είτε μέσω του CLI, είτε του WebApp και πλέον είναι σε κατάλληλη OPenAPI μορφή ώστε να γίνει Import στο Postman.

Στο Postman το import του .yaml αρχείου δημιουργεί ένα API και από αυτό ένα Collection. Το Collection που προκύπτει περιέχει τις ίδιες πληροφορίες με το αρχικό. Το API που προκύπτει είναι πλήρως Documented όπως φαίνεται και παρακάτω:



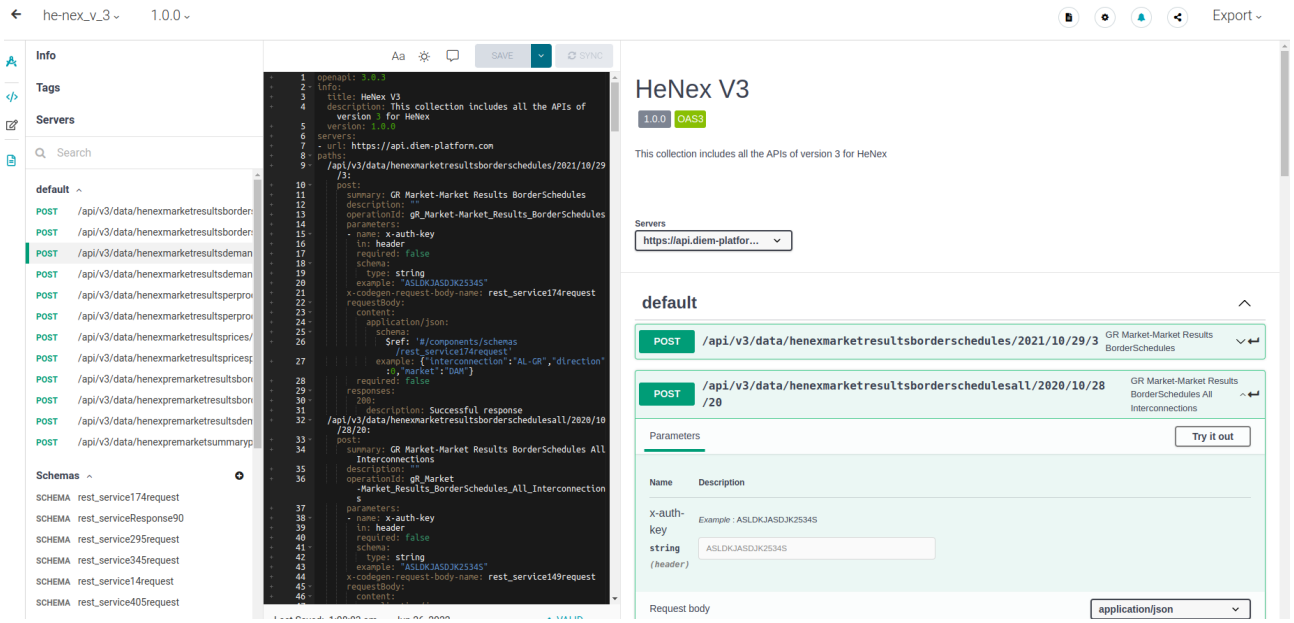
Σχήμα 7.5: *Postman Presentation API Documentation*



Σχήμα 7.6: *Postman Presentation API Documentation Request Body*

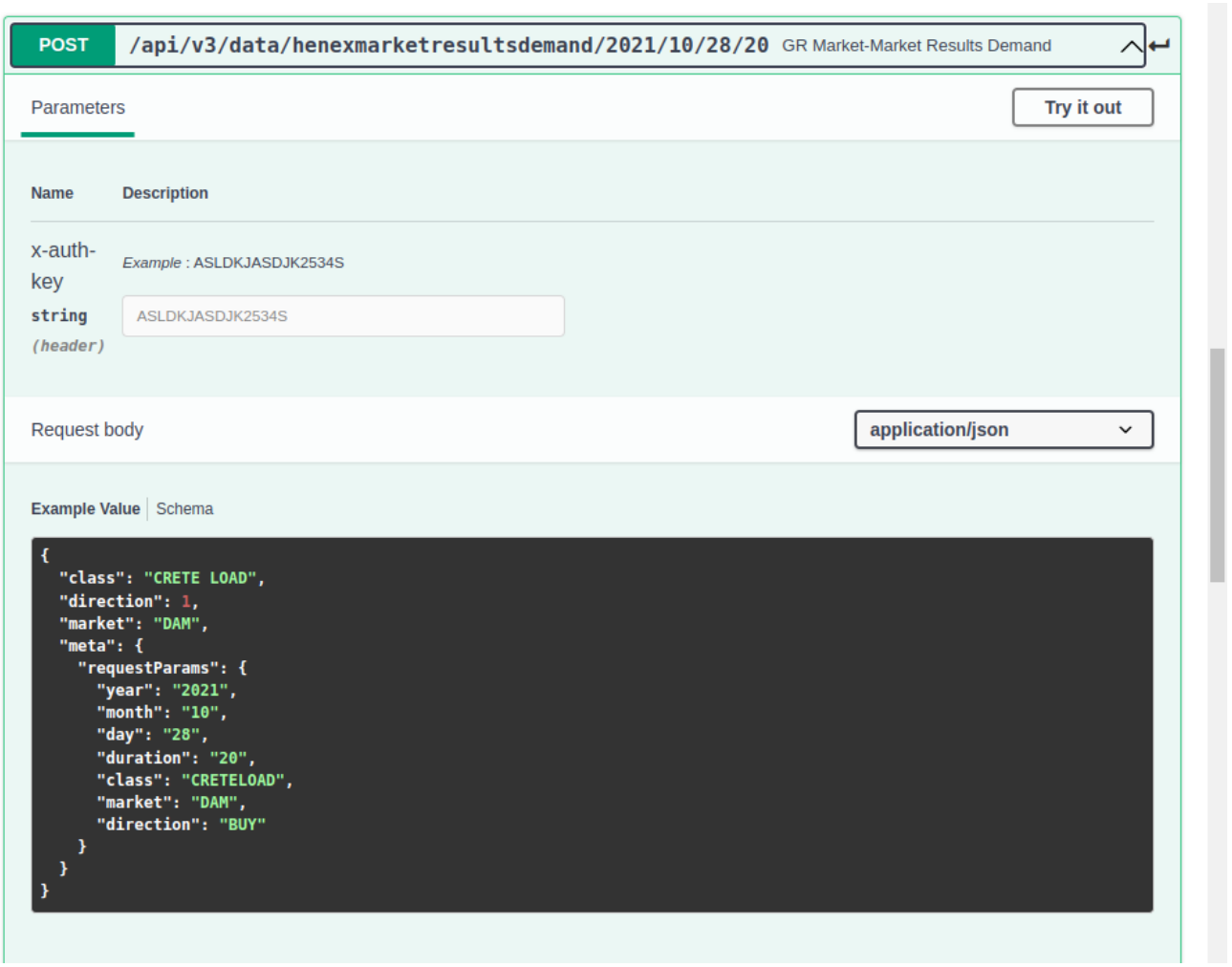
Επιπλέον, το αρχείο που προέκυψε από τη δεύτερη μετατροπή και φορτώσαμε στο Postman, μπορεί να χρησιμοποιηθεί και για το εργαλείο Swagger. Το tool αυτό το δέχεται χωρίς να εμφανίζεται κάποιο error και δημουργεί και αναπαριστά το API όπως φαίνεται παρακάτω:

Η παρακάτω εικόνα περιέχει στα αριστερά τα endpoints του API, στη μέση το OpenAPI (.yaml) Documentatiom που δώθηκε και στα δεξιά έχουν οπτικοπιηθεί οι πληροφορίες του κάθε endpoint, ενώ και υπάρχει και η δυνατότητά δοκιμής τους (κουμπί Try it out).

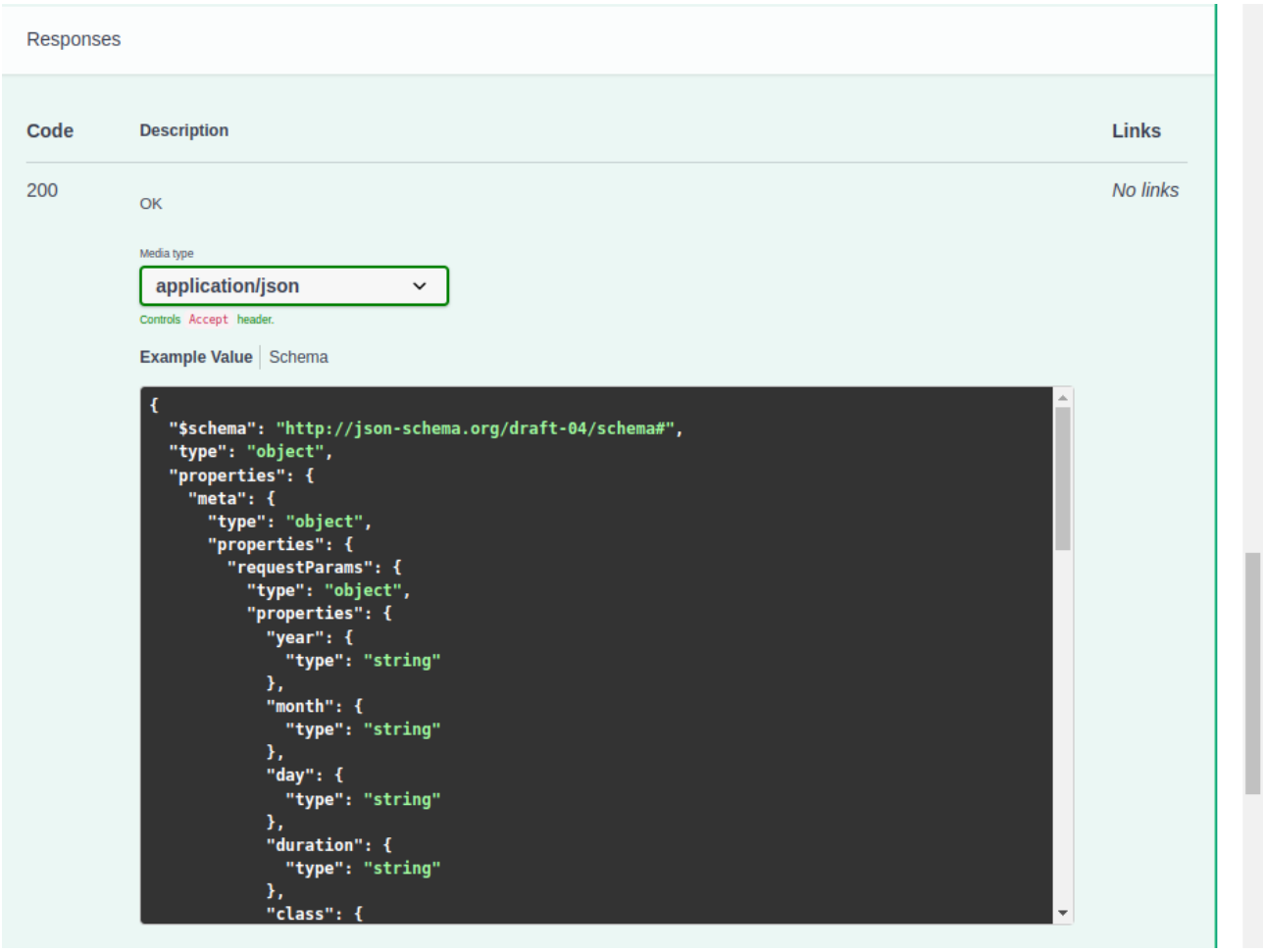


Σχήμα 7.7: *Swagger Presentation API Documentation*

Ακολουθεί η παρουσίαση του UI ενός endpoint που περιέχει και request και response body. Παρατηρούμε ότι μεταφέρονται όλες οι πληροφορίες και χωρίζονται στις αντίστοιχες κατηγορίες.



Σχήμα 7.8: *Swagger Presentation API Documentation*



Σχήμα 7.9: *Swagger Presentation API Documentation*

Κεφάλαιο 8

Παρατηρήσεις και Μελλοντικές Επεκτάσεις

8.1 Συμπεράσματα

Η συγκεκριμένη εργασία παρουσιάζει ουσιαστικά μία κυκλική διαδρομή η οποία ξεκινάει από ένα Documentation ενός Collection, το μετατρέπει σε OpenAPI μορφή και το παρουσιάζει με τη βοήθεια του Visual Paradigm και τέλος καταλήγει στη δημιουργία API και του αρχικού Collection. Συνδυάστηκαν δύο αρκετά γνωστά εργαλεία, το Postman και το Visual Paradigm, ώστε από τη δημιουργία ενός Collection (Postman) να μπορούμε να φτάσουμε εύκολα στην δημιουργία και οπτικοποίησή ενός API (Visual Paradigm) και το αντίστροφο. Καθώς ύστερα από σχετική αναζήτηση δεν βρέθηκαν αντίστοιχα συστήματα, θεωρούμε ότι θα δίνεται πλέον η δυνατότητα κανείς να αναπτύσει και να κατανοεί με πιο εύκολο και αποδοτικό τρόπο ένα API.

8.2 Μελλοντικές Προτάσεις

Οι επεκτάσεις που θεωρούμε ότι θα μπορούσαν να γίνουν αφούν δύο κατηγορίες, τεχνικά ζητήματα και τον τρόπο χρήσης.

Όσον αφορά τα τεχνικά ζητήματα, όπως αναφέρθηκε και παραπάνω, η αναπαράσταση των request και response bodies μέσω κλάσεων στο Visual Paradigm γίνεται με ασύνδετο τρόπο στην περίπτωση nested bodies. Η λύση σε αυτό το ζήτημα θα μείωνε σε κάποιο βαθμό την ανάγκη για προσαρμογή του REST API χειροκίνητα. Ωστόσο, θεωρούμε ότι η μέχρι τώρα αδυναμία οφείλεται σε κάποιο εσωτερικό bug του Visual Paradigm κατά το Reverse OpenAPI, καθώς το αρχείο που εισάγουμε περιέχει την αντίστοιχη πληροφορία. Επιπλέον, θεωρούμε ότι θα ήταν χρήσιμο να υπάρχει δυνατότητα επιλογής συγκεκριμένου REST API για εισαγωγή νέου Documentation πέρα από τη δυνατότητα να εισάγεται σε όλες τις περιπτώσεις το ίδιο.

Όσον αφορά τον τρόπο χρήσης, θεωρούμε ότι θα ήταν πολύ χρήσιμο το Import και Export των αρχείων στο Visual Paradigm να γίνεται με πιο αυτοματοποιημένο τρόπο. Σε αυτό θα βοηθούσε το Command Line Interface του Visual Paradigm [24]. Ωστόσο, δεν δίνεται ακόμα η δυνατότητα για εισαγωγή και εξαγωγή OpenAPI αρχείων.

Βιβλιογραφία

[1] *API Definition*. https://www.grow-digital.gr/ti-einai-ena-api/.

[2] *OpenAPI Definition*. https://support.smartbear.com/swaggerhub/docs/tutorials/openapi-3-tutorial.html.

[3] *Postman Tool*. https://www.postman.com/.

[4] *HTTP Requests*. https://www.tutorialspoint.com/http/http_requests.html.

[5] *Postman Visualizing Respones*. https://learning.postman.com/docs/sending-requests/visualizer/.

[6] *Postman API Builder*. https://learning.postman.com/docs/designing-and-developing-your-api/the-api-workflow/.

[7] *Postman Mpck Servers*. https://learning.postman.com/docs/designing-and-developing-your-api/mocking-data/setting-up-mock/.

[8] *Visual Paradigm*. https://www.visual-paradigm.com/.

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

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

[11] *Sequence Diagram*. https://en.wikipedia.org/wiki/Sequence_diagram.

[12] *Class Diagram*. http://www0.dmst.aueb.gr/louridas/lectures/dais/uml/ar01s05.html.

[13] *Swagger*. https://swagger.io/resources/webinars/getting-started-with-swagger/.

[14] *Javascript*. https://www.javascript.com/.

[15] *Postman to OpenAPI npm package*. https://www.npmjs.com/package/postman-to-openapi.

[16] *Info Object OpenAPI*. https://spec.openapis.org/oas/v3.0.3.html#info-object.

[17] *Nested Object*. https://www.ibm.com/docs/no/db2/11.5?topic=documents-json-nested-objects.

[18] *generate-schema package*. https://www.npmjs.com/package/generate-schema.

[19] *Postman Public API Network Collection*. https://www.postman.com/3497755/workspace/k3r-ck-s-public-workspace/collection/3497755-63a0159e-7bfb-46be-9e80-5e04d8a27723?ctx=documentation.

[20] *chalk npm package*. https://www.npmjs.com/package/chalk.

[21] *figlet npm package*. https://www.npmjs.com/package/figlet.

[22] *Visual Studio Code*. https://code.visualstudio.com/.

[23] *ReactJS*. https://reactjs.org/.

[24] *Command Line Interface Visual Paradigm*. https://www.visual-paradigm.com/support/documents/vpuserguide/124/255_commandlinei.html.