

HERMES

Back-End

System Architecture

Version: 1.2.1

Status: Released

Ιστορικό Αναθεωρήσεων

Date	Version	Description	Authors
<19/01/2021>	1.0	Initial version	Nick Dimitrakopoulos Paraskevi-Theofania Gourgioti Ioannis Christou
<18/02/2021>	1.1	API Reference added	Nick Dimitrakopoulos Paraskevi-Theofania Gourgioti Ioannis Christou
<22/02/2021>	1.2	API Reference edited	Nick Dimitrakopoulos Paraskevi-Theofania Gourgioti Ioannis Christou
<05/07/2021>	1.2.1	Minor updates	Nick Dimitrakopoulos

Πίνακας Περιεχομένων

Ιστορικό Αναθεωρήσεων	2
Πίνακας Περιεχομένων.....	3
Εισαγωγή	4
Σκοπός αυτού του εγγράφου	4
Λίγα λόγια για το Back-End υποσύστημα.....	4
Κύριοι στόχοι του υποσυστήματος	4
Το Back-End ως μέρος του συνολικού ΠΣ	5
Εξαρτήσεις.....	5
Εξαρτώμενες υπηρεσίες	5
Αρχιτεκτονική.....	6
Dependencies	6
Data Model	6
AbstractEntity	6
Application	6
Data Access Layer	7
Exception Handling	7
REST Endpoints	8
API Reference.....	9
Organization Employee Endpoints.....	9
Citizen Endpoints.....	10

Εισαγωγή

Σκοπός αυτού του εγγράφου

Σκοπός αυτού του εγγράφου είναι να τεκμηριωθεί η αρχιτεκτονική του Back-End του Πληροφοριακού Συστήματος (εφεξής ΠΣ) ΕΡΜΗΣ. Απευθύνεται σε μηχανικούς λογισμικού που επιθυμούν να κατανοήσουν, να συντηρήσουν ή να επεκτείνουν το συγκεκριμένο υποσύστημα.

Ορισμός, Ακρωνύμια και Συντομογραφίες

Όρος	Σημασία
Keycloak	Λογισμικό ανοιχτού κώδικα, το οποίο παρέχει Single-Sign-On (SSO) και Identity and Access Management (IAM) δυνατότητες σε μοντέρνες εφαρμογές και υπηρεσίες.

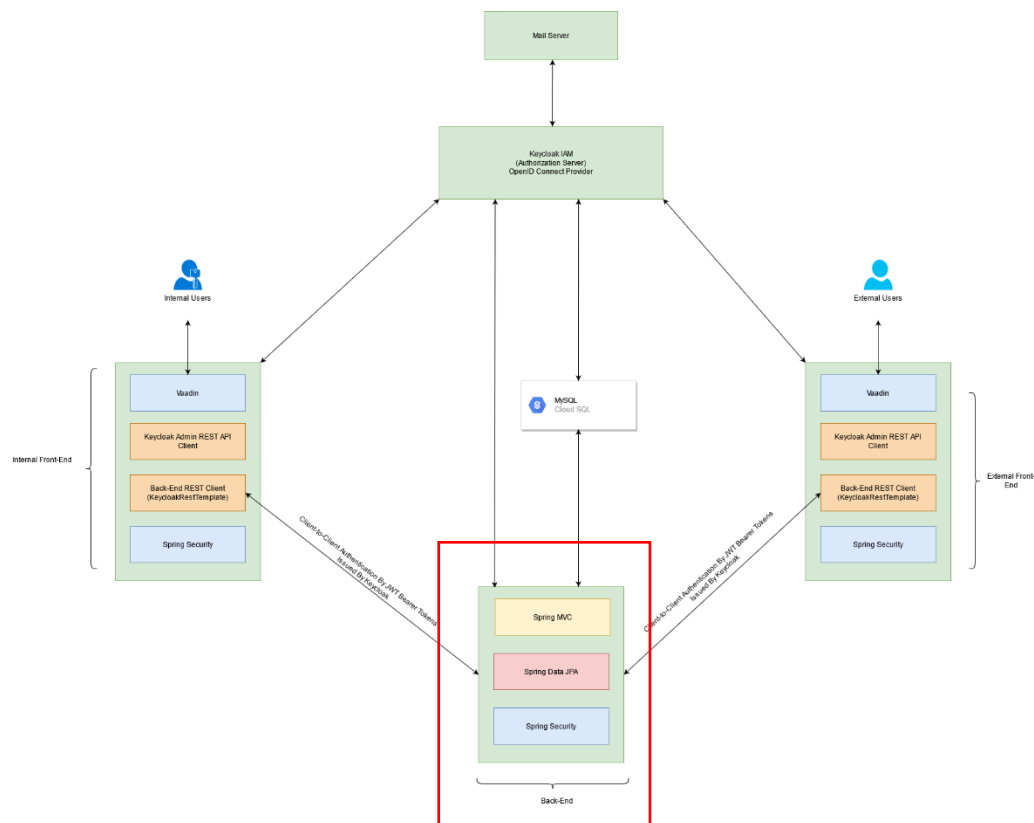
Λίγα λόγια για το Back-End υποσύστημα

Το Back-End υποσύστημα έχει ως στόχο την διαχείριση των αιτήσεων κάθε οργανισμού που είναι ενταγμένος στο ΠΣ ΕΡΜΗΣ. Παρέχει ένα REST API, το οποίο δεν είναι εκτεθειμένο στο δημόσιο ίντερνετ και μπορεί να χρησιμοποιηθεί από οποιοδήποτε άλλο υποσύστημα που έχει ως απαίτηση την παροχή υπηρεσιών management των αιτήσεων.

Κύριοι στόχοι του υποσυστήματος

- Παροχή management υπηρεσιών για τις αιτήσεις στους υπαλλήλους του κάθε οργανισμού.
- Παροχή management υπηρεσιών για τις αιτήσεις στους πολίτες.
- Να παρέχει τις παραπάνω υπηρεσίες με ασφάλεια μέσω του IAM Keycloak, χρησιμοποιώντας Access Control μηχανισμούς όπως:
 - Attribute-based access control (ABAC)
 - Role-based access control (RBAC)
 - User-based access control (UBAC)

Το Back-End ως μέρος του συνολικού ΠΣ



Εξαρτήσεις

Για να θεωρείται διαθέσιμο το Back-End ως υπηρεσία πρέπει να είναι διαθέσιμα και λειτουργικά:

- Ένα instance της **MySQL**
- Ένα instance του **Keycloak**

Εξαρτώμενες υπηρεσίες

Το Back-End παρέχει υπηρεσίες στα ακόλουθα υποσυστήματα του ΠΣ ΕΡΜΗΣ:

- Στο **Internal Front-End**, το οποίο απευθύνεται στους εσωτερικούς χρήστες του ΠΣ.
- Στο **External Front-End**, το οποίο απευθύνεται στους εξωτερικούς χρήστες του ΠΣ.

Αρχιτεκτονική

Dependencies

Το Back-End χρησιμοποιεί το **Spring Boot 2.4.5** ως κύριο framework. Τα κύρια dependencies του είναι τα:

- **Spring Data JPA**
- **Spring Web**
- **Spring Security**
- **Keycloak Client Adapter For Spring Boot (14.0.0)**

Data Model

AbstractEntity

Κάθε entity που υπάρχει ήδη ή πρόκειται να δημιουργηθεί στο μέλλον πρέπει να κάνει extend την abstract κλάση **AbstractEntity**. Η **AbstractEntity** φροντίζει να παρέχει σε κάθε οντότητα τα βασικά πεδία που απαιτούνται για:

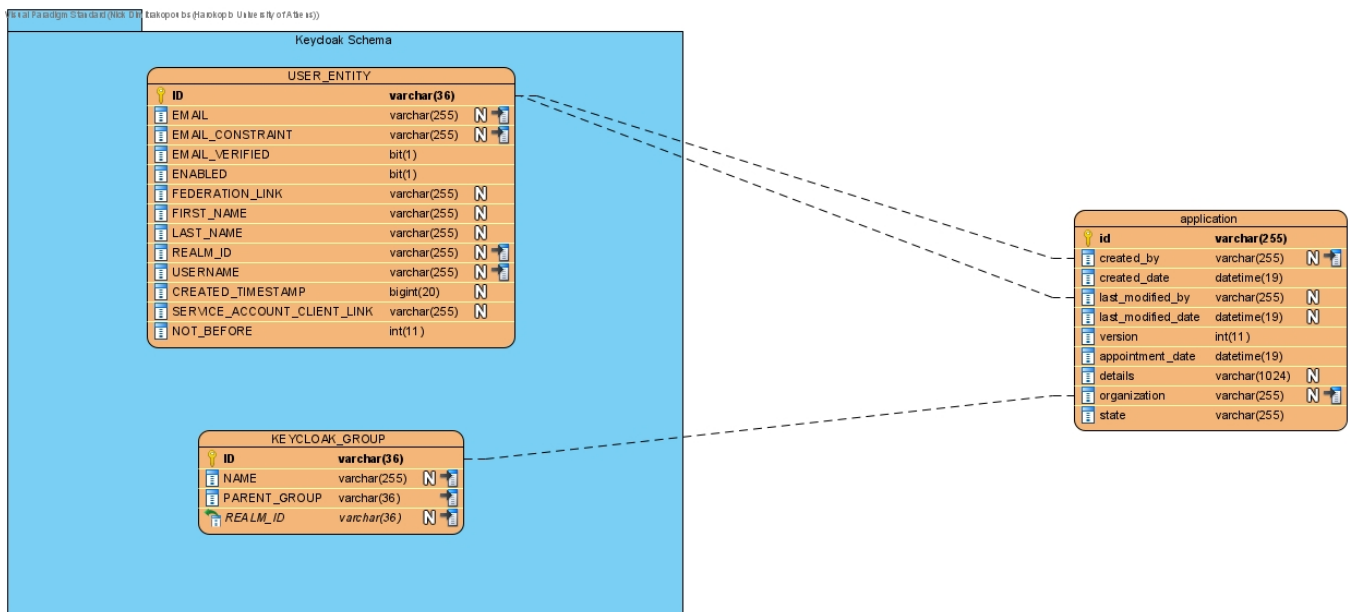
- **Την ταυτοποίηση της οντότητας:** PK τύπου String σε UUID (Universally Unique Identifier) μορφή
- **Βασικό Auditing για κάθε οντότητα** (ποιος/α και πότε δημιούργησε ή τροποποίησε τελευταία φορά την οντότητα. Για περισσότερες πληροφορίες ανατρέξτε [εδώ](#)).
- **Υποστήριξη locking για αποτροπή ανωμαλιών κατά τη ταυτόχρονη πρόσβαση των δεδομένων.** Χρησιμοποιείται Optimistic Locking, για περισσότερες πληροφορίες ανατρέξτε [εδώ](#).

Application

Η οντότητα των αιτήσεων των πολιτών. Κληρονομεί όλα τα χαρακτηριστικά του AbstractEntity και επιπρόσθετα έχει τα εξής γνωρίσματα:

- **ID πολίτη** που κάνει την αίτηση (χρησιμοποιούμε το **created_by** του auditing γιατί μόνο ο πολίτης έχει δικαίωμα δημιουργίας αίτησης)
- **ID οργανισμού** στον οποίο καταθέτεται η αίτηση
- **Κατάσταση της αίτησης** (περιγράφεται στο Functional Requirements έγγραφο)
- **Λεπτομέρειες σχετικά με την αίτηση** (σχόλια)
- **Ημερομηνία και ώρα που επιθυμεί ο πολίτης να γίνει η συνάντηση**

Τα ID πολίτη και ID οργανισμού προέρχονται από το Keycloak. Η πληροφορία μεταδίδεται μέσω του token.



Εικόνα 1 Η έμμεση συσχέτιση του σχήματος του Keycloak με αυτό του Back-End

Data Access Layer

Ως Data Access Layer χρησιμοποιείται το **Spring Data JPA**. Μέσω των **JpaRepository<T,ID>** interfaces μπορούμε να έχουμε τις βασικές μεθόδους για να κάνουμε CRUD operations χωρίς να γράφουμε boilerplate κώδικα.

Η ενσωμάτωση των access control μηχανισμών γίνεται με τη χρήση του **@Query**. Μπορούμε να χρησιμοποιήσουμε **SpEL (Spring Expression Language)** ώστε να παράγουμε queries τα οποία χρησιμοποιούν πληροφορίες από το Access Token του Keycloak.

Για παράδειγμα:

```
@Query("SELECT a FROM Application a WHERE a.createdBy =
?#{authentication.name}")
Page<Application> findAll(Pageable pageable);
```

Αυτό το query φιλτράρει τη σελίδα των αιτήσεων που θα επιστραφεί βάσει το username του χρήστη που έκανε το request. Οποιαδήποτε πληροφορία υπάρχει στο Token μπορεί να χρησιμοποιηθεί στο @Query.

Μπορείτε να ανατρέξετε [εδώ](#) για περισσότερες λεπτομέρειες σχετικά με αυτή τη δυνατότητα.

Exception Handling

Κατά τη διάρκεια της λειτουργίας του Back-End είναι παραπάνω από πιθανό να συμβούν κάποιες αναμενόμενες (ή μη) εξαιρέσεις. Κοινοί τύποι εξαιρέσεων αποτελούν:

- Η μη εύρεση ενός ζητούμενου πόρου (404)
- Αποτυχία στο validation μιας δοσμένης οντότητας (Hibernate Validator)

- Μια συγκρουόμενη ενημέρωση μιας οντότητας (Optimistic Locking)
- Κάποιο απροσδόκητο Internal Server Error (500)

Το Spring Web προσφέρει εξελιγμένο μηχανισμό διαχείρισης σφαλμάτων, ο οποίος μας βοηθάει στο να αποφεύγουμε να γράφουμε boilerplate και unreadable try-catch κώδικα. Οι **@ExceptionHandler** ενεργοποιούνται όποτε πετάγεται η συγκεκριμένη εξαίρεση που έχει δηλωθεί στο value του annotation. Η μέθοδος στην οποία γίνεται annotate ο **@ExceptionHandler** πλέον καθορίζει τι θα επιστραφεί ως απάντηση στο request.

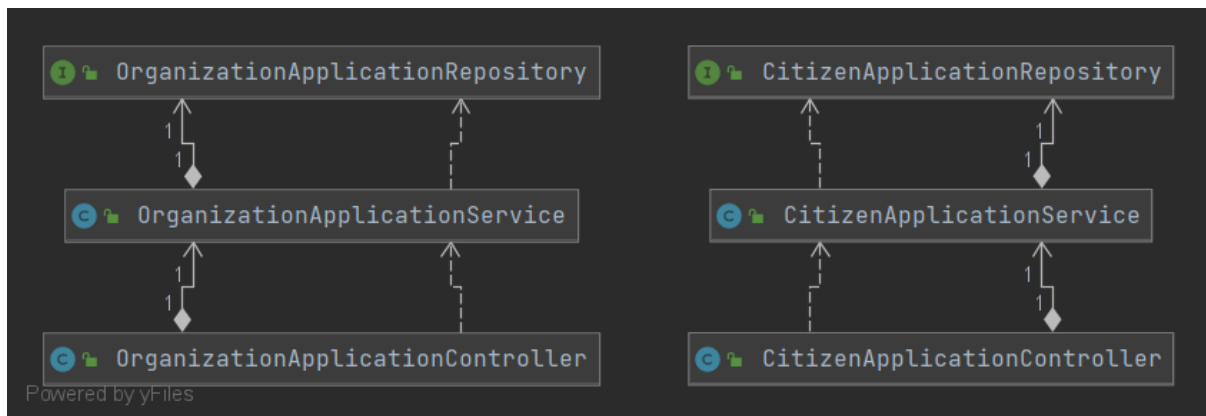
```
//Exception controller when @Valid fails
@ExceptionHandler(MethodArgumentNotValidException.class)
public ResponseEntity<?>
handleValidationException(MethodArgumentNotValidException ex)
{
    List<String> errors = ex.getBindingResult().getAllErrors()
        .stream()
        .map(x-> x.getDefaultMessage())
        .collect(Collectors.toList());

    return ResponseEntity.badRequest().body(Map.of("errors", errors));
}
```

Ένα παράδειγμα εφαρμογής του @ExceptionHandler

REST Endpoints

Για την κατασκευή των endpoints έχει χρησιμοποιηθεί η κλασική προσέγγιση του γνωστού Repository-Service-Controller pattern.



To Repository-Service-Controller Pattern

API Reference

Organization Employee Endpoints

URL:	/organization/application
Method:	GET
Short Description:	Επιστροφή αιτήσεων ραντεβού του οργανισμού που εργάζεται ο υπάλληλος (σελιδοποιημένα)
Authentication / Authorization:	Authentication : Bearer Token Authorization: ROLE_ORG_EMPLOYEE
Headers:	<ul style="list-style-type: none">• Accept: application/json• Authentication: Bearer
Path Parameters:	-
Query Parameters:	<ul style="list-style-type: none">• offset (int)• limit (int)
Body:	-
Response Status Codes:	200 OK 400 Bad Request 401 Unauthorized 403 Forbidden 500 Internal server error
Response Body (Success):	List< Application >

URL:	/organization/application/{id}
Method:	GET
Short Description:	Επιστροφή της αίτησης ραντεβού του οργανισμού που εργάζεται ο υπάλληλος με το δοσμένο id
Authentication / Authorization:	Authentication : Bearer Token Authorization: ROLE_ORG_EMPLOYEE
Headers:	<ul style="list-style-type: none">• Accept: application/json• Authentication: Bearer
Path Parameters:	<ul style="list-style-type: none">• id (string)
Query Parameters:	-
Body:	-
Response Status Codes:	200 OK 400 Bad Request 401 Unauthorized 403 Forbidden 404 Not Found 500 Internal server error
Response Body (Success):	Application

URL:	/organization/application/count
Method:	GET
Short Description:	Επιστροφή πλήθους αιτήσεων του οργανισμού που εργάζεται ο υπάλληλος
Authentication / Authorization:	Authentication : Bearer Token Authorization: ROLE_ORG_EMPLOYEE

Headers:	<ul style="list-style-type: none"> Accept: application/json Authentication: Bearer
Path Parameters:	-
Query Parameters:	-
Body:	-
Response Status Codes:	200 OK 401 Unauthorized 403 Forbidden 500 Internal server error
Response Body (Success):	Long

URL:	/organization/application/{id}
Method:	PUT
Short Description:	Ενημέρωση της αίτησης ραντεβού του οργανισμού που εργάζεται ο υπάλληλος με το δοσμένο id.
Authentication / Authorization:	Authentication : Bearer Token Authorization: ROLE_ORG_EMPLOYEE
Headers:	<ul style="list-style-type: none"> Accept: application/json Content-Type: application/json Authentication: Bearer
Path Parameters:	<ul style="list-style-type: none"> id (string)
Query Parameters:	-
Body:	-
Response Status Codes:	200 OK 400 Bad Request 401 Unauthorized 403 Forbidden 500 Internal server error
Response Body (Success):	-

Citizen Endpoints

URL:	/citizen/application
Method:	GET
Short Description:	Επιστροφή αιτήσεων του πολίτη που κάνει την αίτηση (σελιδοποιημένα)
Authentication / Authorization:	Authentication : Bearer Token Authorization: ROLE_CITIZEN
Headers:	<ul style="list-style-type: none"> Accept: application/json Authentication: Bearer
Path Parameters:	-
Query Parameters:	<ul style="list-style-type: none"> offset (int) limit (int)
Body:	-
Response Status Codes:	200 OK 400 Bad Request 401 Unauthorized 403 Forbidden 500 Internal server error

Response Body (Success):	List< Application >
--------------------------	-------------------------------------

URL:	/citizen/application/{id}
Method:	GET
Short Description:	Επιστροφή της αίτησης ραντεβού του πολίτη με το δοσμένο id
Authentication / Authorization:	Authentication: Bearer Token Authorization: ROLE_CITIZEN
Headers:	<ul style="list-style-type: none"> Accept: application/json Authentication: Bearer
Path Parameters:	<ul style="list-style-type: none"> id (string)
Query Parameters:	-
Body:	-
Response Status Codes:	200 OK 400 Bad Request 401 Unauthorized 403 Forbidden 404 Not Found 500 Internal server error
Response Body (Success):	Application

URL:	/citizen/application/count
Method:	GET
Short Description:	Επιστροφή πλήθους αιτήσεων του πολίτη που έκανε την αίτηση.
Authentication / Authorization:	Authentication : Bearer Token Authorization: ROLE_CITIZEN
Headers:	<ul style="list-style-type: none"> Accept: application/json Authentication: Bearer
Path Parameters:	-
Query Parameters:	-
Body:	-
Response Status Codes:	200 OK 401 Unauthorized 500 Internal server error
Response Body (Success):	Long

URL:	/citizen/application
Method:	POST
Short Description:	Καταχώρηση νέας αίτησης ραντεβού του οργανισμού που εργάζεται ο υπάλληλος με το δοσμένο id.
Authentication / Authorization:	Authentication : Bearer Token Authorization: ROLE_CITIZEN
Headers:	<ul style="list-style-type: none"> Accept: application/json Content-Type: application/json Authentication: Bearer
Path Parameters:	<ul style="list-style-type: none"> id (string)
Query Parameters:	-
Body:	-
Response Status Codes:	200 OK

	400 Bad Request 401 Unauthorized 403 Forbidden 500 Internal server error
Response Body (Success):	-

URL:	/organization/application/{id}
Method:	PUT
Short Description:	Ενημέρωση της αίτησης του πολίτη που έκανε την αίτηση με το δοσμένο id
Authentication / Authorization:	Authentication : Bearer Token Authorization: ROLE_CITIZEN
Headers:	<ul style="list-style-type: none"> • Accept: application/json • Content-Type: application/json • Authentication: Bearer
Path Parameters:	<ul style="list-style-type: none"> • id (string)
Query Parameters:	-
Body:	-
Response Status Codes:	200 OK 400 Bad Request 401 Unauthorized 403 Forbidden 500 Internal server error
Response Body (Success):	-