Τεχνική Αναφορά — MyCityGov

1. Σκοπός και περιγραφή προβλήματος

Η εφαρμογή **MyCityGov** αποτελεί μία διαδικτυακή πύλη ψηφιακών αιτημάτων και ραντεβού με τον Δήμο. Στόχος είναι η εξυπηρέτηση των πολιτών μέσω υποβολής αιτημάτων/αιτήσεων, η παρακολούθηση της πορείας τους και ο προγραμματισμός ραντεβού με τις δημοτικές υπηρεσίες, σε ένα κατανεμημένο περιβάλλον με ασφαλές REST API και ενσωμάτωση εξωτερικών υπηρεσιών.

2. Ρόλοι χρηστών

Η εφαρμογή υποστηρίζει τους παρακάτω ρόλους:
• **Πολίτης (CITIZEN)**: εγγραφή, σύνδεση, υποβολή αιτήματος, παρακολούθηση κατάστασης, προγραμματισμός ραντεβού.
• **Υπάλληλος (EMPLOYEE)**: προβολή αιτημάτων του τμήματος, ανάληψη αιτήματος, ενημέρωση κατάστασης, ενημέρωση κατάστασης ραντεβού του τμήματος.
• **Διαχειριστής (ADMIN)**: πλήρης πρόσβαση σε αιτήματα και ραντεβού, διαχείριση τύπων αιτημάτων και ωραρίων ραντεβού ανά τμήμα.

3. Αρχιτεκτονική και τεχνολογίες

Η εφαρμογή ακολουθεί **layered architecture** με καθαρό διαχωρισμό:
• **Web/UI layer**: Spring MVC + Thymeleaf templates.
• **Service layer**: επιχειρησιακή λογική και κανόνες.
• **Repository layer**: Spring Data JPA.

Χρησιμοποιούνται επίσης έννοιες **Hexagonal Architecture** (Ports/Adapters) για εξωτερικές υπηρεσίες:
• `GovAuthPort` και `PhoneNumberPort` ως συμβόλαια.
• `GovAuthPortImpl` και `PhoneNumberPortImpl` ως adapters.

Τεχνολογίες:
• Spring Boot 3, Spring MVC, Spring Security
• JPA/Hibernate
• H2 (file-based)
• Springdoc OpenAPI / Swagger UI
• JWT (stateless για REST API)

Δομή πακέτων (ενδεικτικά):
• `web/ui`: Controllers και Thymeleaf views (UI).
• `web/api`: REST controllers για API clients.
• `core/service`: επιχειρησιακή λογική και κανόνες.
• `core/repository`: JPA repositories.
• `core/model`: οντότητες και enums.
• `core/security`: authentication/authorization και JWT.
• `core/port`: συμβόλαια προς εξωτερικές υπηρεσίες.

4. Μοντέλο δεδομένων (σύνοψη)

Βασικές οντότητες:
• **User**: στοιχεία πολίτη/υπαλλήλου/διαχειριστή.
• **ServiceDepartment**: τμήματα δήμου + ωράριο ραντεβού.
• **RequestType**: τύποι αιτημάτων ανά τμήμα.
• **Request**: αιτήματα πολιτών (πρωτόκολλο, κατάσταση, SLA κ.λπ.).
• **Appointment**: ραντεβού πολιτών με τμήματα και κατάσταση.

5. Επιχειρησιακοί κανόνες

• Κάθε αίτημα αποκτά μοναδικό αριθμό πρωτοκόλλου και SLA (προθεσμία) ανάλογα με τον τύπο αιτήματος.
• Η κατάσταση ενός αιτήματος δεν μπορεί να επιστρέψει από **Ολοκληρωμένο** ή **Απορρίφθηκε**.
• Ραντεβού επιτρέπονται μόνο εντός του ωραρίου του τμήματος.
• Υπάλληλος μπορεί να διαχειριστεί αιτήματα μόνο του τμήματός του, ενώ ο διαχειριστής έχει πλήρη πρόσβαση.

6. Κύριες λειτουργικότητες

Πολίτης

• Εγγραφή με έλεγχο τηλεφώνου από εξωτερική υπηρεσία.
• Υποβολή αιτήματος με τύπο/θέμα/περιγραφή.
• Παρακολούθηση κατάστασης αιτήματος.
• Προγραμματισμός ραντεβού εντός επιτρεπόμενων ωρών.

Υπάλληλος

• Πρόσβαση στα αιτήματα του τμήματος.
• Ανάληψη αιτήματος και ενημέρωση κατάστασης/σχολίων.
• Ενημέρωση κατάστασης ραντεβού του τμήματος.

Διαχειριστής

• Πλήρης πρόσβαση σε όλα τα αιτήματα/ραντεβού.
• Διαχείριση τύπων αιτημάτων.
• Ορισμός ωραρίων ραντεβού ανά τμήμα.

7. Επιβεβαίωση και validation

• Bean Validation σε DTOs (π.χ. ημερομηνία ραντεβού στο μέλλον).
• Έλεγχοι ακεραιότητας (μοναδικότητα email/ΑΦΜ/ΑΜΚΑ/τηλεφώνου).
• Χειρισμός σφαλμάτων με κατάλληλα μηνύματα προς τον χρήστη.
• Έλεγχος ορίων ωραρίου ανά τμήμα πριν από την καταχώριση ραντεβού.

8. Ασφάλεια

• **UI**: Stateful authentication (form login, cookie-based, Spring Security).
• **API**: Stateless JWT tokens μέσω `/api/auth/tokens`.
• **Roles**: CITIZEN / EMPLOYEE / ADMIN.

9. REST API & Swagger

• REST endpoints για αιτήματα, ραντεβού, τύπους αιτημάτων, authentication.
• Τεκμηρίωση μέσω **OpenAPI** και **Swagger UI**.

Χρήσιμα endpoints:
• `GET /v3/api-docs`
• `GET /swagger-ui.html`

10. REST API (σύνοψη λειτουργιών)

• **Auth**: έκδοση JWT token για API clients.
• **Requests**: λίστα/λεπτομέρειες/δημιουργία, ανάθεση και ενημέρωση κατάστασης.
• **Appointments**: λίστα/δημιουργία ραντεβού.
• **Request Types**: λίστα ενεργών τύπων αιτημάτων.

Ενδεικτικά endpoints:
• `POST /api/auth/tokens`
• `GET /api/requests` / `GET /api/requests/{id}`
• `POST /api/requests` / `POST /api/requests/{id}/assign`
• `PUT /api/requests/{id}/status`
• `GET /api/appointments` / `POST /api/appointments`
• `GET /api/request-types`

11. Εξωτερική υπηρεσία (NOC)

Η εφαρμογή καταναλώνει την εξωτερική υπηρεσία NOC (black-box) για:
1) **Gov Login**
   - `POST /api/v1/gov/login`
   - Request: `{ afm, pin }`
   - Response: `{ token, expiresAt, citizen { afm, amka, fullName } }`

2) **Phone validation**
   - `GET /api/v1/phone-numbers/{raw}/va-lidations`
   - Response: `{ raw, valid, type, e164 }`

Χειρισμός σφαλμάτων:
• Invalid credentials → εμφανίζεται σχετικό μήνυμα.
• Μη διαθέσιμη υπηρεσία → εμφανίζεται μήνυμα σφάλματος και αποτυγχάνει η ενέργεια.

12. Ωράρια ραντεβού

• Προεπιλογή ανά τμήμα: **09:00 – 17:00**.
• Τα ραντεβού επιτρέπονται μόνο μέσα στο ωράριο του αντίστοιχου τμήματος.
• Διαχειριστής μπορεί να αλλάξει τα ωράρια ανά τμήμα.

13. Ροές χρήσης (ενδεικτικά)

1) **Πολίτης**: εγγραφή → σύνδεση → δημιουργία αιτήματος → παρακολούθηση κατάστασης.
2) **Υπάλληλος**: είσοδος → προβολή αιτημάτων τμήματος → ανάληψη → ενημέρωση κατάστασης/σχολίων.

3) **Διαχειριστής**: πρόσβαση σε όλα τα αιτήματα → ενημέρωση κατάστασης → διαχείριση τύπων → ορισμός ωραρίων.
4) **Ραντεβού**: επιλογή υπηρεσίας → επιλογή ημερομηνίας/ώρας → έλεγχος ωραρίου → καταχώριση.

14. Εκτέλεση εφαρμογής

1) Εκκινήστε την εξωτερική υπηρεσία NOC (απαιτείται για registration και gov login).
2) Εκκινήστε την εφαρμογή:

```
./mvnw spring-boot:run
```

Η εφαρμογή τρέχει στο: `http://localhost:8080`

Χρήσιμες διευθύνσεις:
• Swagger UI: `http://localhost:8080/swagger--ui.html`
• H2 Console: `http://localhost:8080/h2-cons-ole`

15. Περιορισμοί και επεκτάσεις

• Η αποθήκευση αρχείων υλοποιείται ως URL (δεν υπάρχει πραγματικό upload).
• Δεν έχει υλοποιηθεί ξεχωριστή υπηρεσία εξωτερικού πελάτη (SPA).
• Δεν υπάρχει live ειδοποίηση (email/SMS), μόνο επιχειρησιακή ροή.