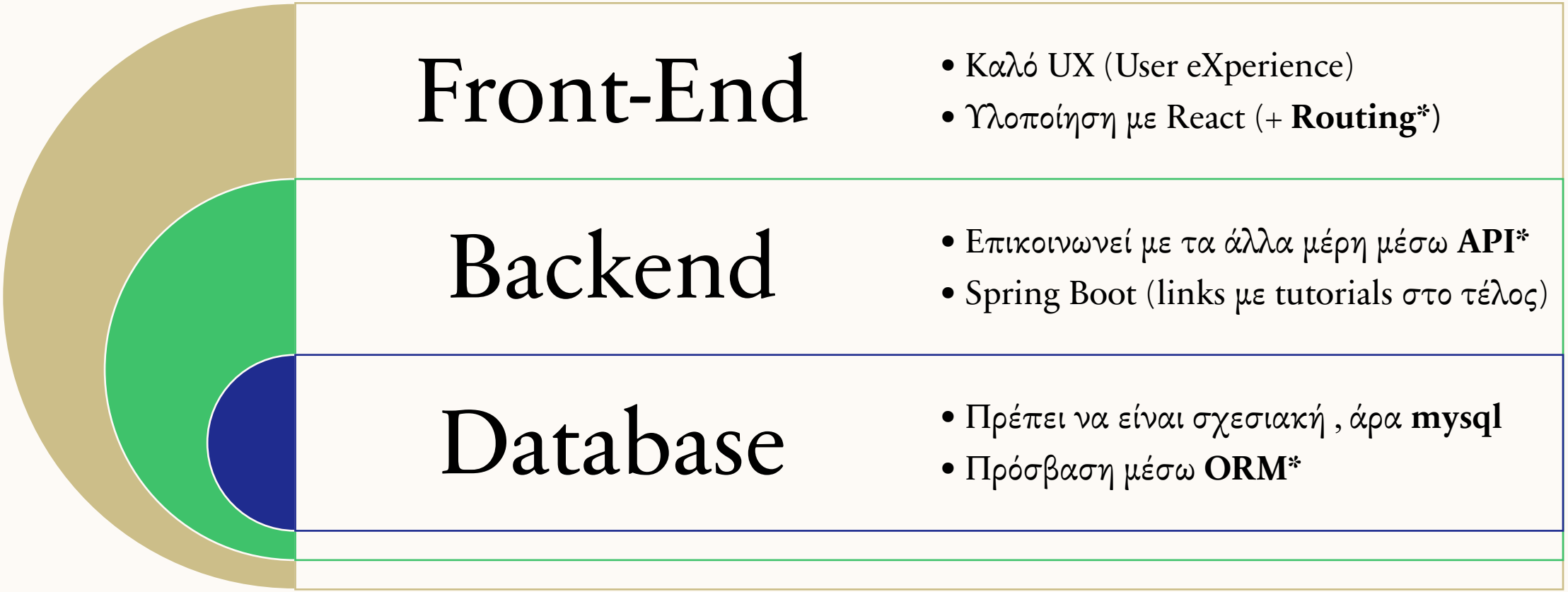
ΕΡΓΑΣΙΑ
ΕΙΔΙΚΑ ΘΕΜΑΤΑ ΤΕΧΝΟΛΟΓΙΑΣ ΛΟΓΙΣΜΙΚΟΥ

ΣΥΝΟΠΤΙΚΑ

1. Αντικείμενο Εργασίας
2. Μεθοδολογία Ανάπτυξης
3. Repo , Αποθετήριο
4. Αρχιτεκτονική Συστήματος
5. Authorization / Authentication
6. Testing
7. Tech Stack

1. ΑΝΤΙΚΕΙΜΕΝΟ ΕΡΓΑΣΙΑΣ



(*) Επεξηγήσεις σε επόμενες διαφάνειες

2. ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΠΤΥΞΗΣ

- Δήλωση ομάδων έως 09/11
- Πρέπει να ακολουθήσουμε **Agile/Scrum*** μεθοδολογία, όπου:
  - Οι απαιτήσεις είναι σε μορφή **user stories***
  - Θα υπάρχει **product backlog***
  - Ένα **board*** για κάθε **sprint***
  - Τα sprints έχουν διάρκεια 15 ημερών

  (Καλό είναι κάθε user story να μπορεί να γίνει σε ένα sprint)

SCRUM

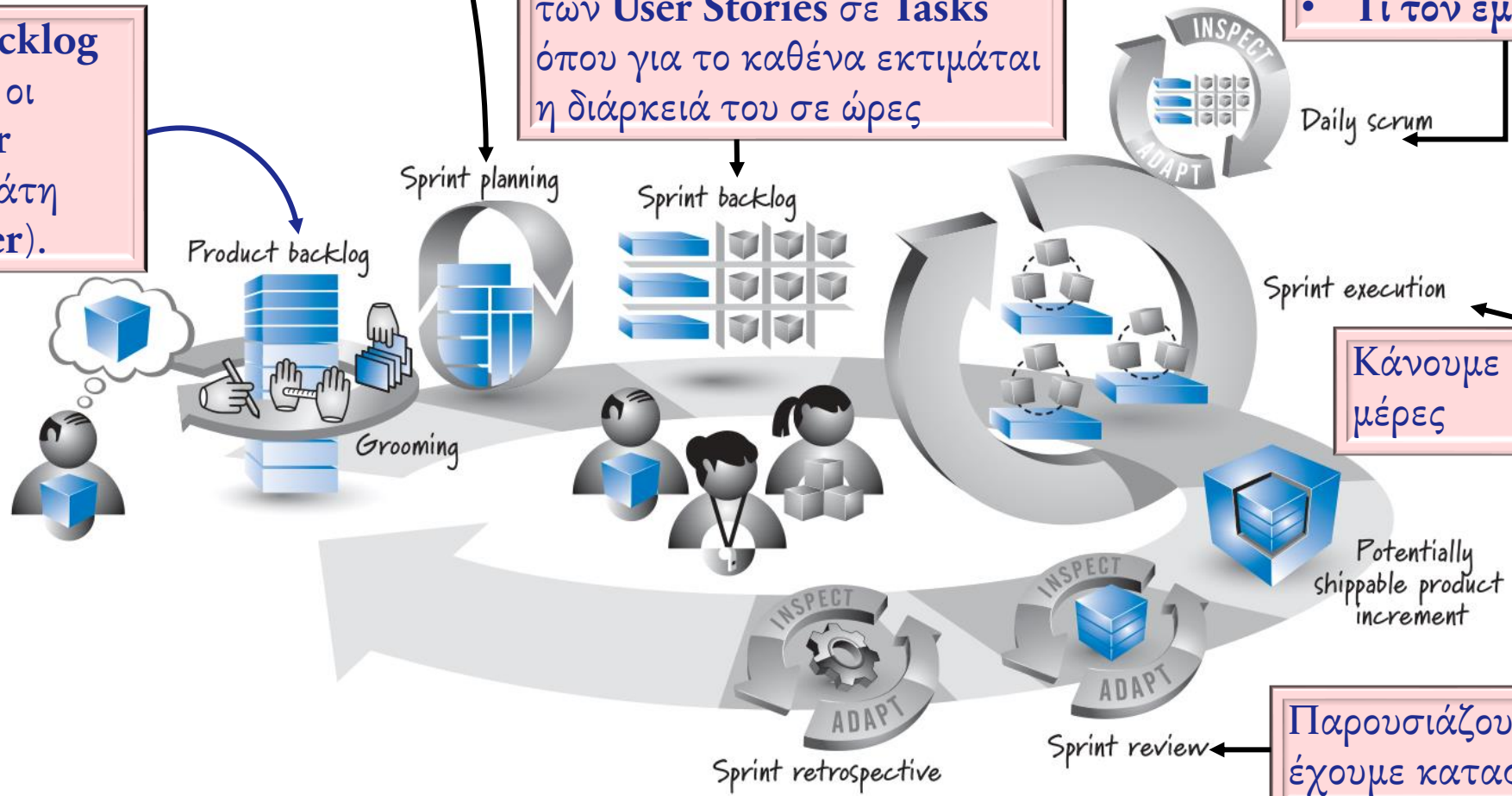
Στο **Sprint Planning** ο **Product Owner** επιλέγει ποια στοιχεία του **backlog** θέλει να ολοκληρωθούν στο **τρέχον sprint** δημιουργώντας έτσι το **sprint backlog**

Στο **Product backlog** μαζεύονται όλες οι απαιτήσεις (**user stories**) του πελάτη (**Product Owner**).

Το **sprint backlog** ουσιαστικά είναι η ανάλυση των **User Stories** σε **Tasks** όπου για το καθένα εκτιμάται η διάρκειά του σε ώρες

Στο **Daily Scrum** ο καθένας απαντάει:

- **Τι έκανε ?**
- **Τι θα κάνει ?**
- **Τι τον εμποδίζει ?**



Κάνουμε δουλειά για 15 μέρες

Παρουσιάζουμε τι έχουμε καταφέρει

ΜΕΘΟΔΟΛΟΓΙΑ

1° Βήμα

- Κατανοούμε το σύστημα
- Φτιάχνουμε τα **user stories**

2° Βήμα

- Συλλέγουμε τα **user stories** σε ένα **backlog** (**Jira**)
- Και αρχίζουμε το **sprint planning**

3° Βήμα

- Επιλέγουμε τα μεγαλύτερης προτεραιότητας **user stories**
- Και εκτελούμε **sprints** θέτοντας τα **sprint goals** πριν την έναρξή τους

USER STORIES

- Τα **user stories** αποτελούνται από τρία μέρη:

1. **Περιγραφή:** Σύντομο περιεκτικό κείμενο που απαντάει σε τρείς βασικές ερωτήσεις
   - **Who (user role)**
   - **What (goal)**
   - **Why (reason)**

   As a [user role] I want to [goal] so I can [reason]

   Παράδειγμα:

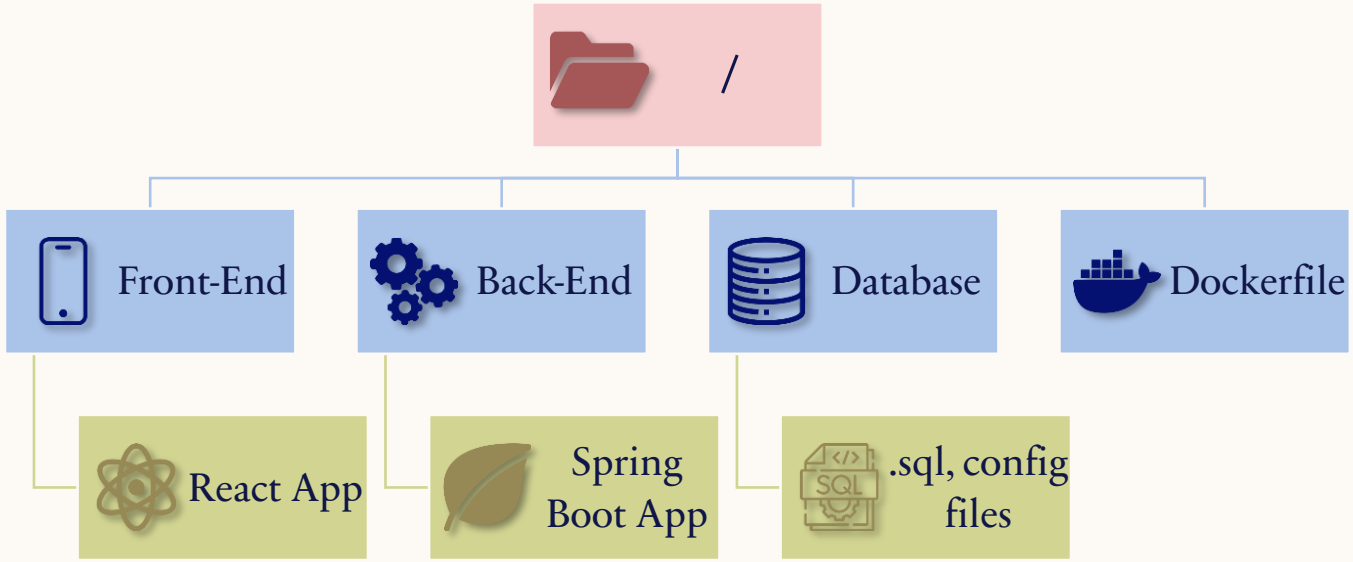
   As a **registered user** I want to **log in** so I can **access subscriber-only content**
2. **Συζήτηση / Λεπτομέρειες**
   - Επιπλέον πληροφορίες που απορρέουν από την συζήτηση με τον πελάτη.
3. **Επιβεβαίωση**
   - Ποιοι έλεγχοι θα διεξαχθούν για να επιβεβαιώσουμε ότι η ιστορία έχει υλοποιηθεί όπως πρέπει

3. REPOSITORY

GitHub (Public ή Private ?) και Git

Project Structure (Παράδειγμα):

(διευκολύνεται με αυτό τον τρόπο και η τελική μετατροπή σε docker app)



Icons source: Freepik - Flaticon.com

4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

Front-End

- Επικοινωνεί με το business logic μέσω **Restful web services**
- Θα είναι σχεδιασμένο ως **SPA** (Single Page Application)
- Θα χρειαστεί βιβλιοθήκη **routing** για το **React**. Π.χ. **React Router**, κτλπ.

Back-End

- Αντ/φής γλώσσα, όπως **Java**
- **Spring Boot**, **Spring Security**
- **API** σε **3 layers** (**controllers**, **business logic**, **data**) (θα δούμε παραδείγματα μετά)
- **Dependency Injection** (Επεξήγηση & Παράδειγμα σε επόμενη διαφάνεια)

4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

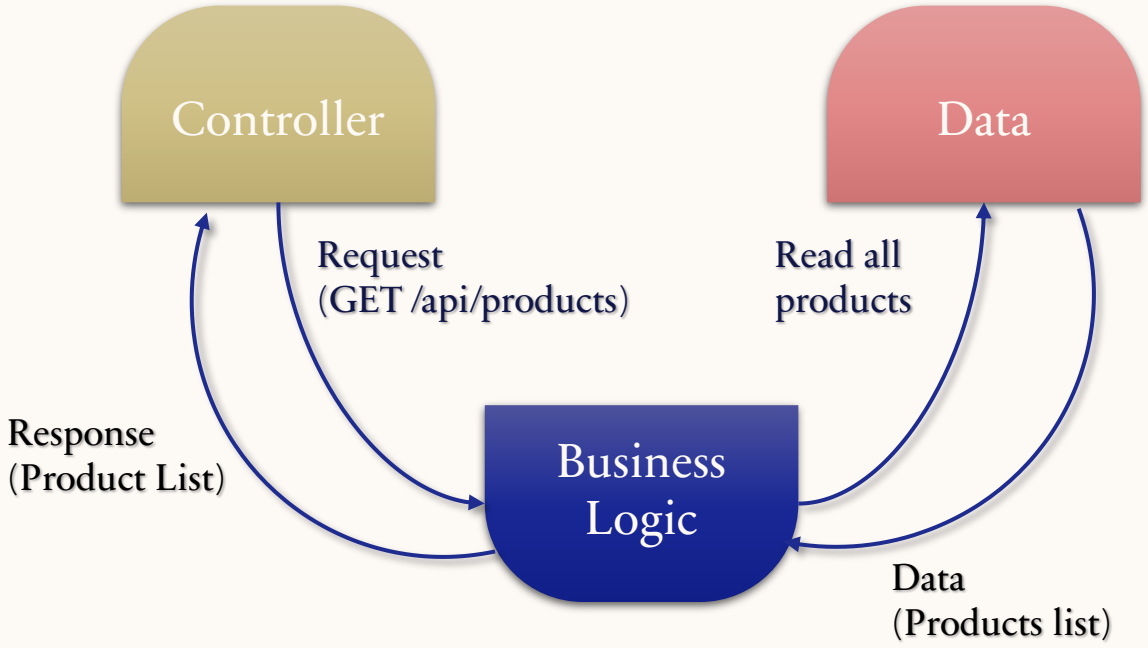
Database

- Σχεσιακή, άρα **mysql** (**MariaDB, MySQL**)
- Επικοινωνία με business logic μέσω **ORM*** (Object Relational Model) (**Hibernate**)
- Θα φτιάξουμε πρώτα την βάση , και θα μπει στον server που έχω ώστε να μην χρειάζεται το κάθε μέλος της ομάδας να την φτιάχνει κι ενημερώνει συνέχεια.

Development

- **Front-End & Back-End** -> **Local**, το κάθε μέλος στο **δικό του Workstation**
- **Database** -> **Remote**, στον server, θα φτιαχτούν **αντίστοιχοι users**
- **Server** -> **Open 24/7** , Στο τέλος θα προσπαθήσουμε να **ανεβάσουμε την εργασία** σε αυτόν μέσω **docker.**

4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ



Τι είναι το **API** σε 3 **layers ?**

Ουσιαστικά αυτό που ζητείται είναι ο **standard** τρόπος με τον οποίο φτιάχνονται **CRUD εφαρμογές**. Έχουμε τα 3 layers:

- **Controller**
  - **Δέχεται τα αιτήματα** (**requests**) από το Front-End **και στέλνει** πίσω τις **απαντήσεις** (**responses**)
- **Business Logic (Service)**
  - **Αποφασίζει τι πρέπει να γίνει με βάση το αίτημα.**
- **Data Layer (ή αλλιώς Repository)**
  - Επικοινωνεί με τη βάση δεδομένων. Εκτελεί **CRUD** μέσω ORM (Hibernate).

4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ - ΠΑΡΑΔΕΙΓΜΑ

Controller

```
@RestController
@RequestMapping("/api/products")
public class ProductController {
  private final ProductService service;
  public ProductController(ProductService service) {
    this.service = service;
  }
  @GetMapping // GET
  public List<Product> getAllProds() {
    return service.findAll();
  }
}
```

4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ - ΠΑΡΑΔΕΙΓΜΑ

Business Logic (Service)

```
@Service
public class ProductService {
  private final ProductData repo; // ORM
  public ProductService(ProductData repo) {
    this.repo = repo;
  }
  public List<Product> findAll() {
    return repo.findAll();
  }
}
```

4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ - ΠΑΡΑΔΕΙΓΜΑ

Data (ORM) - Repository

```
@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {
}

@Entity
public class Product { // Άμεση συσχέτιση με τον πίνακα Product της ΒΔ
    @Id @GeneratedValue
    private Long id;
    private String name;
    private double price;
}
```

5. AUTHORIZATION / AUTHENTICATION

Authorization

Ελέγχω ουσιαστικά ποιες λειτουργίες είναι διαθέσιμες στον συγκεκριμένο χρήστη

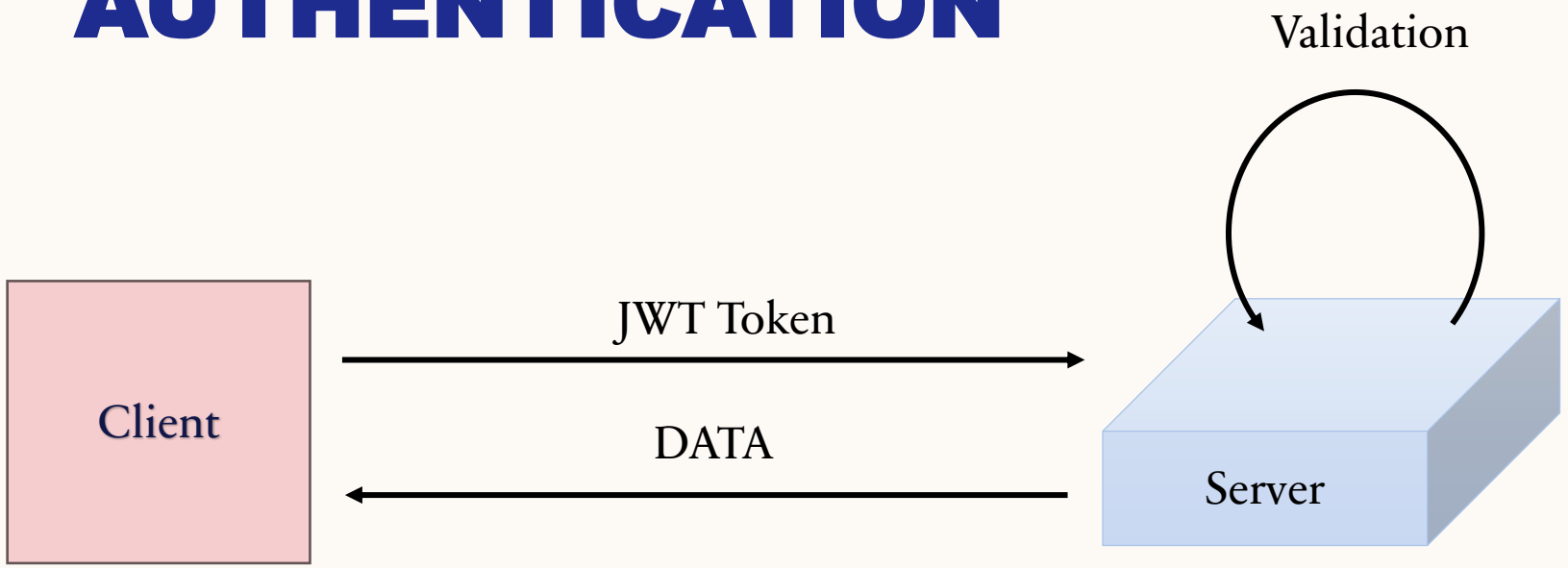
Authentication

Επιβεβαιώνω την ταυτότητα του χρήστη με τα στοιχεία που μου δίνει

Info

JWT (JSON Web Token): Ουσιαστικά μετά το **Authentication** παράγεται ένα **JWT token** μέσω του οποίου θα παρέχεται πρόσβαση σε προστατευμένα **REST Endpoints** και μπορεί να συνδυαστεί με ποια endpoints θα έχει πρόσβαση ο κάθε ρόλος μέσω του **RBAC (Role Based Access Control**

5. AUTHORIZATION / AUTHENTICATION

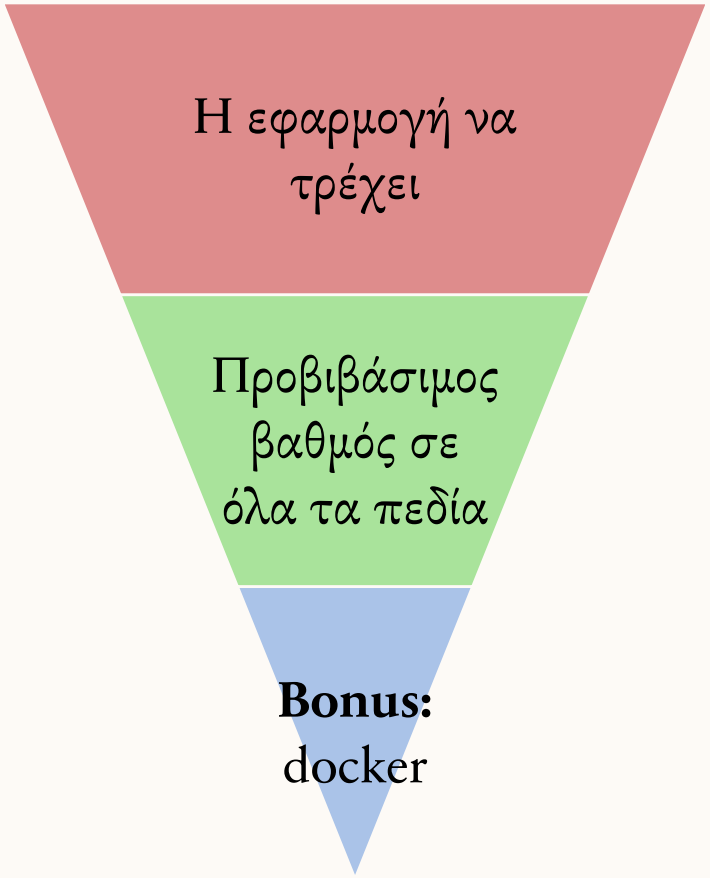


What is JWT? (~5mins)

6. TESTING

- Μόνο κάποια integrated end-to-end tests στο τέλος του εξαμήνου. Θα τα δούμε πιο μετά.

7. ΑΞΙΟΛΟΓΗΣΗ



Πεδίο	Μονάδες
Ύπαρξη και χρήση αποθετηρίου κώδικα	2
Ύπαρξη εργαλείου για την παρακολούθηση των sprints και χρήση αυτού	2
Χρήση Dependency Injection	1
Χρήση ORM	1
Integration Tests	2
Μορφή κώδικα / UI/UX	2

THANK YOU

Valentino Velchev
[Your phone]
[Your email]