***Доклад за дипломен проект***

**Изготвил: Милен Станоев**

**Група: CHC 14222**

**Дипломен ръководител: Георги Стоев**

1. **Цел на проекта и очакван резултат**

Основната цел на дипломния проект беше разработването на **уеб базирана система за управление на бележки**, изградена върху архитектура от микроуслуги. Потребителите трябва да могат да се регистрират, влизат в профила си, създават, редактират и изтриват лични бележки, като същевременно се гарантира сигурността и защитата на информацията.

Очакваният резултат включва:

* Съвременна, сигурна и лесно разширяема система, подходяща както за индивидуална употреба, така и за бъдещи cloud-базирани разширения.
* Микроуслуга за управление на потребителите;
* Микроуслуга за управление на бележки;
* API Gateway за централизирана комуникация;
* Сигурност чрез JWT токени;
* Реактивен уеб интерфейс (React) с Bootstrap за потребителска работа;
* Администраторски панел;
* Мониторинг чрез Prometheus и Spring Boot Actuator
* Пълна Docker-контейнеризация за лесно стартиране и внедряване.

1. **Анализ на темата – актуалност, практическа ценност и възможности за реализация**

Темата е изключително актуална, тъй като архитектурата с микроуслуги и cloud-native приложенията са все по-предпочитани при изграждане на мащабируеми и лесно поддържащи се системи. Управлението на бележки е универсален сценарий, който може да бъде доразвит до платформи за продуктивност, екипни приложения, блогове или дори учебни платформи.

Системата има висока **практическа стойност**:

* При разработката ѝ може да се усвои на работата с REST API и JWT сигурност;
* Помага да се разбере как разделянето на логиката на отделни микроуслуги съдейства за разработката, менажирането и поддръжката на големи системи;
* Може да бъде използвана като основа за реална DevOps/Cloud среда чрез Prometheus.
* Може да бъде надградена до платформа за екипна продуктивност;
* Подходяща е за обучение по DevOps, CI/CD, Docker и Kubernetes;
* Съвместима е с cloud-базирани решения като AWS, Azure, GCP.

1. **Стъпки за изработване на проекта – последователност, използвани ресурси и софтуер**

**Стъпки на реализация:**

1. **Планиране на архитектурата** – Eureka, User service, Notes service, Gateway, frontend.
2. **Изграждане на backend микросървиси**:
   * Регистрация, логин, управление на потребители;
   * CRUD операции върху бележки, асоциирани с потребителя;
   * Защита чрез JWT и Spring Security;
3. **Сигурност и роли** – внедряване на JWT и защита на endpoint-и с @PreAuthorize.
4. **Изграждане на API Gateway,** който маршрутизира заявките**;**
5. **React Frontend,** свързан с backend чрез Axios и JWT**;**
6. **Администраторски панел за управление на потребители**;
7. **Мониторинг** чрез Spring Actuator + Prometheus;
8. **Контейнеризация с Docker и docker-compose** – всички компоненти стартират с една команда.

**Използвани технологии:**

* **Java 17, Spring Boot 3, Spring Security, Spring Cloud Eureka;**
* **MongoDB** като база данни;
* **React.js + Bootstrap** за потребителски интерфейс;
* **JWT** за сигурност;
* **Docker + Docker Compose** за контейнеризация;
* **Prometheus** и **Actuator** за мониторинг;
* **Postman** и **Axios** за API тестове и връзка между frontend и backend.
* **IntelliJ IDEA** като среда за разработка**.**

1. **Трудности по време на работата – как са били превъзмогнати и дали са довели до промени**

Някои от основните предизвикателства бяха:

* **Конфигурация на Eureka и Discovery** – необходимо беше ръчно конфигуриране на всяка микроуслуга и правилна регистрация към Eureka.
* **Проблеми с JWT в gateway-a и микросървисите** – решени чрез централизирана обработка на токените във филтър.
* **Проблеми с ролите и ограниченията по endpoint-и** – решени чрез @PreAuthorize и конфигурация на SecurityFilterChain.
* **Интеграция между frontend и backend** – внимателен контрол върху токените, 403/404 грешки и cross-origin заявки.
* **Създаване на устойчив docker-compose** – включва Eureka, Mongo, всички микросървиси и frontend.

Всички проблеми бяха успешно разрешени и доведоха до по-завършен и стабилен проект.

1. **Изводи – достигнати резултати, подобрения и варианти за разширяване на проекта**

**Проектът постигна всички първоначални цели:**

* **Напълно функционален потребителски и бележков микросървис**;
* **Централизирана Eureka регистрация** и лесно откриване на услуги;
* **Сигурност чрез JWT и Spring Security**;
* **Работещ администраторски панел**;
* **Модерен React интерфейс с Bootstrap**;
* Интегрирани **мониторинг метрики** чрез Actuator и Prometheus;
* **Готова контейнеризация чрез Docker Compose**.

**Възможности за разширяване:**

* Имейл известия и напомняния;
* Управление на файлове (качване и теглене);
* Версионаризиране и статистики за потребителски действия;
* Интернационализация (многоезичност);
* Добавяне на Kafka или RabbitMQ за комуникация между ивентите;
* CI/CD pipeline с GitHub Actions и автоматичен деплой;
* Деплой в Kubernetes и cloud среда (напр. AWS ECS/EKS).