Общие требования к лабораторным

Описание

- Java + `Spring` Boot
- MVC архитектура
- `Maven`
- Git + `GitHub`
- API сервер
- Поэтапное развитие, постоянные коммиты. Если будет всё в один день, на защите будет тяжело оправдаться
- `PostgreSQL`
- Stateless + `JWT` токены в Cookies
- Экспорт отчетов и хранение в `MinIO`
- Полное покрытие тестами
- Полное покрытие логами
- Документация в `Swagger`

Модели

```
public class User {
    private Long id;
    private String username;
    private String password;
    private boolean enabled;
    private Set<Role> roles;
}

public class Role {
    private Long id;
    private String title;
    private Set<Permission> permissions;
}

public class Permission {
    private Long id;
    private String title;
}
```

□ API

- `CRUD` на все сущности
- Создать коллекцию запросов в `Postman` для симуляции работы

□ Доп

- Обработка кастомных исключений
- Кастомная валидация
- уведомления в `Телеграм-бот`
- `RedPanda` очередь
- `Docker`
- Запуск с `Jenkins`

Отчёт по работе должен содержать:

- Текст задания.
- Диаграмма классов разработанной программы.
- Ссылка на public GitHub репозиторий (внутри sql dump + readme.md)
- Рабочее приложение для показа или видео
- Выводы по работе.

□ Проект №1: Кроссплатформенная система управления обновлениями приложений

□ Цель

- Автоматизировать управление версиями и обновлениями кроссплатформенного приложения.

□ Описание

Система отслеживает:

- Какие версии приложения используются на разных платформах
- Поддержка обновлений: принудительное, рекомендованное, отложенное
- Статистика установки обновлений (сколько пользователей перешли на новую версию)
- Уведомления: "Ваша версия устарела — обновитесь!"

□ Модели

```
public class AppVersion {
    private String version;
    private Platform platform;
    private LocalDateTime releaseDate;
    private String changelog;
    private UpdateType updateType;
    private boolean isActive;
}

public class UserDevice {
    private String userId;
    private Platform platform;
    private String currentVersion;
    private LocalDateTime lastSeen;
}

public class UpdateStatsDTO {
    private String version;
    private Map<Platform, Integer> usersCount;
    private Double globalUpdateRate;
}

public enum UpdateType {
    MANDATORY, OPTIONAL, DEPRECATED
}
```

□ API

- POST /api/versions – добавить новую версию
- GET /api/versions/latest?platform=android – последняя версия для платформы
- GET /api/update/check?userId=123&current=1.1.0&platform=ios – проверка обновления
- POST /api/update/log – лог установки обновления
- GET /api/stats/updates – статистика перехода на новые версии

□ Доп

- Тепловая карта распространения версий по платформам и времени
- Загрузка списка устаревших версий из YAML (с датами блокировки)