ИТОГОВОЕ ЭКЗАМЕНАЦИОННОЕ ЗАДАНИЕ
«Развертывание микросервисного приложения управления пользователями в Kubernetes»

Разработать приложение со следующими функциональными возможностями (могут быть другие, но количество функциональных возможностей должно быть не меньше):

1. **Окно авторизации** – пользователи должны иметь возможность войти в систему с помощью логина и пароля
2. **Главная страница** – должна содержать информацию о приложении и ссылки на авторизацию или регистрацию
3. **Личная страница пользователя** – после успешного входа пользователи должны перенаправляться на свою личную страницу, где они смогут видеть информацию о своем профиле
4. **Наличие базы данных** – приложение должно использовать базу данных для хранения логинов и паролей пользователей. (*Опционально*: Ведение в базе данных списка пользователей с их состояниями (например, активный, неактивный, заблокированный)).

Посещаемость	Оценка	Теория
Пропусков более 50%	Автоматическое снижение оценки на один балл от полученной по практической части	– Необходимо устно ответить на 3 теоретических вопроса – Для получения оценки необходимо правильно ответить как минимум на 2 из 3 вопросов
Пропусков от 25% до 50%	Оценка без изменений при работающем приложении и ответе на вопросы по практической части	Необходимо устно ответить на 1 теоретический вопрос
Пропусков менее 25%	Оценка без изменений при работающем приложении	Не нужно отвечать на теоретические вопросы
Пропусков 0%	Оценка «удовлетворительно» может быть получена без выполнения практического задания *ИЛИ* При выполнение практического задания возможно частичное упрощение конфигурации	Не нужно отвечать на теоретические вопросы

На оценку «удовлетворительно»

1. Kubernetes Deployment для базового приложения:
   - Создать манифесты Kubernetes для развертывания **монолитного приложения** и базы данных
   - Использовать **минимум 2 Pod** для приложения
   - Использовать **1 Pod** для базы данных
2. Базовые сервисы:
   - **Сервис веб-приложения** – обеспечивает доступ к фронтенду (тип: **NodePort** для внешнего доступа)
   - **Сервис базы данных** – обеспечивает доступ к БД из приложения (тип: **ClusterIP**)
3. Базовая конфигурация:
   - Создать **ConfigMap** для хранения нефункциональных данных приложения
4. Обязательные требования:
   - Все ресурсы разместить в **отдельном Namespace** user-platform-exam
   - Добавить **метки (labels)** ко всем ресурсам
   - Использовать **Dockerfile** для создания образа приложения

На оценку «хорошо»

1. Микросервисная архитектура:
Разделить монолитное приложение на микросервисы (минимум **3 отдельных сервиса**). (могут быть другие, но количество должно быть не меньше 3): :

- **Сервис аутентификации (auth-service)** – отвечает за обработку входа пользователей, регистрацию новых аккаунтов, валидацию сессий
- **Сервис профиля (profile-service)** – управляет данными профиля пользователя
- **Сервис уведомлений (notification-service)** – отвечает за отправку уведомлений пользователям

2. Обеспечение устойчивости приложения

- Использовать **Liveness и Readiness Probes** для всех Deployment
  - LivenessProbe: HTTP GET на путь /health/live
  - ReadinessProbe: HTTP GET на путь /health/ready
  - Параметры: initialDelaySeconds: 30, periodSeconds: 10
- Настроить **минимум 2 реплики** для каждого сервиса

3. Безопасное хранение конфиденциальных данных. Использовать **Secret** для хранения:

4. Динамическая конфигурация интерфейса

- Использовать **ConfigMap** для изменения текстов интерфейса
- *Пример: создать ConfigMap ui-config с данными:*
  LOGIN_TITLE: "Вход в систему"
  REGISTER_TITLE: "Регистрация"
  WELCOME_MESSAGE: "Добро пожаловать в платформу"

На оценку «отлично»

1. Система отчетов и фоновые задачи:
Добавить **четвертый сервис**:

- **Сервис отчетов (report-service)** – генерирует отчеты на основе данных пользователей

2. Автоматизированные фоновые задачи
Реализовать **3 CronJob**:

- **CronJob 1: Сбор ежедневной статистики**
  Пример
  - *Название: daily-stats-collector*
  - *Расписание: "0 2 * * *" (каждый день в 2:00)*
  - *Задача: собирает статистику по пользователям за последние 24 часа*
- **CronJob 2: Отправка уведомлений**
  Пример
  - *Название: notification-sender*
  - *Расписание: "0 9 * * *" (каждый день в 9:00)*
  - *Политика конкурентности: Forbid*
- **CronJob 3: Очистка данных**
  Пример
  - *Название: data-cleanup*
  - *Расписание: "0 0 * * 0" (каждое воскресенье)*
  - *Задача: логирует операцию очистки устаревших данных*

3. Управление развертыванием через Helm

- Создать **Helm Chart** для всего приложения
- Параметризовать настройки:
  - Количество реплик каждого сервиса
  - Версии образов
  - Расписание CronJob