

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Омский государственный технический университет»
Кафедра «Автоматизация системы обработки информации
и управления»

Отчет

по дисциплине:

«Современные инструментальные средства разработки программного
обеспечения»

Выполнил:

студент группы ПИН-201

Покотило Д.А.

Подпись: _____

Проверил:

ст. преподаватель Кабанов

А.А.

Подпись: _____

Омск 2023

Автоматизированная система управления операционными картами на предприятии АО «Высокие технологии»

Описание

Однако, я могу предоставить общую информацию о том, как автоматизированная система управления операционными картами может функционировать на предприятии.

Автоматизированная система управления операционными картами может включать в себя следующие элементы:

1. Система мониторинга и сбора данных:

- Сбор данных о текущем состоянии операций и процессов на предприятии.
- Мониторинг ключевых показателей производительности.

2. Визуализация данных:

- Представление операционных данных в виде графиков, диаграмм и операционных карт для наглядного отображения текущего статуса и производительности.

3. Управление ресурсами:

- Оптимизация распределения ресурсов на основе данных об операционных процессах.
- Автоматизация управления запасами и инвентаризацией.

4. Управление задачами и проектами:

- Отслеживание выполнения задач и проектов на предприятии.
- Распределение задач между сотрудниками и контроль сроков выполнения.

5. Система отчетности:

- Генерация отчетов о производственной деятельности и эффективности.
- Анализ данных для выявления трендов и улучшения операционных процессов.

6. Интеграция с другими системами:

- Взаимодействие с другими информационными системами предприятия для обмена данными.

7. Безопасность данных:

- Обеспечение безопасности и конфиденциальности операционных данных.

8. Обучение и развитие:

- Использование системы для обучения персонала и оптимизации производственных процессов.

Реализация автоматизированной системы управления операционными картами может значительно повысить эффективность работы предприятия, улучшить принятие решений и обеспечить более гибкое реагирование на изменения в бизнес-среде.

< 3 / 3 >    

Стек технологий создания автоматизированной системы управления операционными картами на предприятии



ChatGPT

Создание автоматизированной системы управления операционными картами требует использования различных технологий, в зависимости от конкретных требований предприятия и характера бизнес-процессов. Вот некоторые ключевые технологии, которые могут быть применены в стеке технологий для такой системы:

1. Базы данных:

- Выбор подходящей базы данных для хранения операционных данных. Это может быть реляционная база данных (например, PostgreSQL, MySQL, Microsoft SQL Server) или NoSQL-решение (например, MongoDB для гибкости в хранении неструктурированных данных).

2. Веб-разработка:


- Использование веб-технологий для создания пользовательского интерфейса. Это может включать в себя HTML, CSS, JavaScript для фронтенда. Для бэкенда можно выбрать языки программирования, такие как Python (Django, Flask), Java (Spring), Node.js или Ruby on Rails.

3. Фреймворки для разработки приложений:

- Использование специализированных фреймворков для разработки приложений управления данными, например, Django, Flask, Ruby on Rails.

4. Облачные технологии:

- Размещение приложения в облаке может обеспечить гибкость, масштабируемость и доступность. Популярные облачные платформы включают Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP).

5. **Микросервисная архитектура:**
 - Разделение функциональности системы на небольшие, автономные микросервисы. Это облегчает масштабирование, обновление и поддержку приложения.
 6. **API (интерфейсы программирования приложений):**
 - Разработка API для взаимодействия между различными компонентами системы и интеграции с другими приложениями и службами.
 7. **Системы управления версиями:**
 - Использование систем управления версиями, таких как Git, для отслеживания изменений в коде и совместной работы разработчиков.
 8. **Бизнес-аналитика и обработка данных:**
 - Инструменты для анализа данных, построения отчетов и визуализации, такие как Tableau, Power BI, Apache Superset.
 9. **Безопасность:**
 - Внедрение средств безопасности, включая шифрование данных, аутентификацию и авторизацию, чтобы обеспечить защиту от несанкционированного доступа.
 10. **Инструменты тестирования:**
 - Использование средств автоматизированного тестирования для проверки функциональности и стабильности системы.
 1. **Искусственный интеллект (по необходимости):**
 - Интеграция технологий машинного обучения и анализа данных для оптимизации операций и предсказания трендов.
- 
- Выбор конкретных технологий зависит от требований вашего предприятия,

Для начала разработки необходимо выполнить следующие шаги:



ChatGPT

Создание автоматизированной системы управления операционными картами на предприятии включает в себя несколько этапов. Вот общие шаги, которые могут потребоваться для реализации такой системы:

1. Анализ бизнес-процессов:

- Идентификация ключевых операционных процессов на предприятии, которые нужно отслеживать и управлять через операционные карты.

2. Определение требований:

- Составление подробного списка требований к системе на основе анализа бизнес-процессов. Это включает в себя функциональные и нефункциональные требования.

3. Выбор технологического стека:

- Решение о технологиях, которые будут использоваться для разработки системы. Выбор баз данных, языков программирования, фреймворков и других технологий.

4. Проектирование базы данных:

- Разработка структуры базы данных для хранения операционных данных. Определение связей между различными сущностями и атрибутами.

5. Разработка пользовательского интерфейса:

- Создание интерфейса системы для взаимодействия с пользователями. Разработка страниц операционных карт, отображения данных и возможности взаимодействия.



взаимодействия.

6. **Разработка бизнес-логики:**
 - Написание кода для обработки бизнес-логики системы, включая механизмы сбора данных, обработки запросов и генерации отчетов.
7. **Реализация микросервисной архитектуры (по необходимости):**
 - Если требуется масштабируемость и гибкость, разделение функциональности на микросервисы может быть эффективным подходом.
8. **Разработка API:**
 - Создание API для обеспечения взаимодействия между различными компонентами системы и интеграции с другими приложениями.
9. **Реализация системы управления доступом:**
 - Внедрение механизмов аутентификации и авторизации для обеспечения безопасности данных и ограничения доступа к операционным картам.
10. **Тестирование:**
 - Проведение тестирования системы на различных уровнях (юнит-тестирование, интеграционное тестирование, системное тестирование) для обеспечения стабильной работы и соответствия требованиям.
11. **Внедрение:**
 - Постепенное внедрение системы на предприятии, обучение персонала и обеспечение поддержки в процессе внедрения.
12. **Обучение и документация:**
 - Подготовка обучающих материалов и документации для пользователей системы.
13. **Поддержка и сопровождение:**
 - Обеспечение регулярного обновления, поддержки и дальнейшего развития системы в соответствии с потребностями предприятия.

14. Мониторинг и оптимизация:

- Внедрение механизмов мониторинга для отслеживания производительности системы и внесение оптимизаций при необходимости.

Каждый этап требует внимательного планирования, сотрудничества между различными отделами предприятия и внимания к деталям.



Описание дизайна

Дизайн автоматизированной системы управления операционными картами включает в себя несколько ключевых аспектов, таких как пользовательский интерфейс, база данных, бизнес-логика и безопасность. Ниже представлено описание дизайна такой системы:

1. Пользовательский интерфейс (UI):

- **Главная страница:**
 - Персонализированный дашборд с обзором ключевых операционных показателей.
 - Быстрый доступ к операционным картам различных процессов.
- **Операционные карты:**
 - Интерактивные и интуитивно понятные операционные карты с возможностью динамического изменения параметров.
 - Графическое отображение статуса операций, задач и ресурсов.
- **Фильтрация и поиск:**
 - Механизмы фильтрации данных для быстрого нахождения нужной информации.
 - Поиск по ключевым словам, тегам или временным интервалам.

2. Бизнес-логика:

- **Управление бизнес-процессами:**
 - Модульная структура для управления различными бизнес-процессами через операционные карты.
 - Автоматизация выполнения задач и событий в зависимости от статуса операций.
- **Расширяемость и гибкость:**
 - Возможность легкого добавления новых операционных карт и адаптации системы под изменяющиеся потребности предприятия.
 - Поддержка микросервисной архитектуры для модульности и масштабируемости.

3. База данных:

- **Структура данных:**
 - Разработка оптимальной структуры базы данных для хранения информации об операционных картах, ресурсах и их состояниях.
 - Использование реляционных или NoSQL подходов в зависимости от требований.
- **Интеграция данных:**
 - Обеспечение интеграции с другими системами предприятия через единый механизм хранения данных.
 - Защита данных с использованием механизмов шифрования и управления доступом.

4. Безопасность:

- **Управление доступом:**
 - Реализация системы управления доступом с уровнями привилегий для различных ролей пользователей.
 - Аудит и мониторинг защиты доступа и изменений в данных.
- **Шифрование данных:**
 - Применение шифрования данных в покое и в передаче для защиты конфиденциальности.
 - Защита от атак и утечек данных с использованием современных методов шифрования.

5. Интеграция с внешними системами:

- **API и веб-сервисы:**
 - Разработка API для интеграции с другими системами предприятия.
 - Поддержка стандартных протоколов (REST, SOAP) для обмена данными.

6. Тестирование и обеспечение качества:

- **Тестирование по уровням:**
 - Проведение тестирования на различных этапах разработки: от юнит-тестов до системных тестов.
 - Тестирование производительности для оптимизации работы системы.

7. Мониторинг и оптимизация:

- **Механизмы мониторинга:**
 - Внедрение системы мониторинга для отслеживания производительности и стабильности системы.
 - Анализ данных мониторинга для выявления узких мест и оптимизации.

Данный дизайн представляет собой общий обзор основных компонентов и принципов, которые могут быть использованы при разработке автоматизированной системы управления операционными картами. Он должен быть адаптирован под конкретные потребности и особенности предприятия.

2 / 2

Примеры кодов



ChatGPT

Разработка полной автоматизированной системы управления операционными картами требует значительного объема кода, который может быть написан на различных языках программирования и использовать разнообразные технологии. Ниже приведен простой пример кода на Python с использованием веб-фреймворка Flask для создания простой системы управления операционными картами.

```
python Copy code  
  
from flask import Flask, render_template, request, redirect, url_for  
import sqlite3  
  
app = Flask(__name__)  
  
# Подключение к базе данных SQLite  
conn = sqlite3.connect('operational_cards.db')  
cursor = conn.cursor()  
  
# Создание таблицы операционных карт, если её нет  
cursor.execute('''  
    CREATE TABLE IF NOT EXISTS operational_cards (  
        id INTEGER PRIMARY KEY AUTOINCREMENT,  
        title TEXT NOT NULL,  
        description TEXT,  
        status TEXT
```

```

    )
'''
conn.commit()

# Закрытие соединения с базой данных
conn.close()

# Маршрут для отображения списка операционных карт
@app.route('/')
def index():
    conn = sqlite3.connect('operational_cards.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM operational_cards')
    cards = cursor.fetchall()
    conn.close()
    return render_template('index.html', cards=cards)

# Маршрут для добавления новой операционной карты
@app.route('/add_card', methods=['POST'])
def add_card():
    conn = sqlite3.connect('operational_cards.db')
    cursor = conn.cursor()
    title = request.form['title']
    description = request.form['description']
    status = request.form['status']
    cursor.execute('INSERT INTO operational_cards (title, description,
conn.commit()
conn.close()
return redirect(url_for('index'))

```

```

if __name__ == '__main__':
    app.run(debug=True)

```

Этот код создает простое веб-приложение с использованием Flask, которое позволяет добавлять операционные карты в базу данных SQLite и отображать список всех карт. Он использует HTML-шаблон (например, с помощью Jinja2) для отображения данных.

Примечание: Этот пример предназначен только для иллюстративных целей и не включает в себя все необходимые элементы для создания полноценной системы управления операционными картами. Реальная система требует более сложной структуры, включая обработку ошибок, систему аутентификации и безопасности, а также другие функциональности.



Еще один пример кода



Приведенный ниже пример кода написан на языке программирования Java с использованием веб-фреймворка Spring Boot. Данный код представляет простую систему управления операционными картами с использованием базы данных H2 для иллюстрации основных концепций. В реальной системе вам, возможно, потребуется использовать более мощную базу данных и реализовать дополнительные функциональности, такие как безопасность, аутентификация и др.

1. Создайте проект:

Используйте вашу любимую среду разработки или систему сборки (например, Maven или Gradle) для создания проекта с поддержкой Spring Boot. Зависимости для базы данных H2 и Spring Web можно добавить в файл `pom.xml` для Maven:

```
xml Copy code

<!-- Добавьте зависимости в файл pom.xml -->
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>
</dependencies>
```

2. Создайте модель для операционных карт:

```
java Copy code

// OperativeCard.java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class OperativeCard {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;
    private String description;
    private String status;

    // геттеры и сеттеры
}
```

3. Создайте репозиторий для работы с базой данных:

```
java Copy code

// OperativeCardRepository.java
import org.springframework.data.jpa.repository.JpaRepository;

public interface OperativeCardRepository extends JpaRepository<OperativeCard> {
    // Дополнительные методы могут быть добавлены по необходимости
}
```

4. Создайте контроллер для взаимодействия с операционными картами:

```
java Copy code

// OperativeCardController.java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class OperativeCardController {

    @Autowired
    private OperativeCardRepository operativeCardRepository;

    @GetMapping("/")
    public String index(Model model) {
        model.addAttribute("cards", operativeCardRepository.findAll());
        return "index";
    }

    @PostMapping("/add_card")
    public String addCard(OperativeCard card) {
        operativeCardRepository.save(card);
        return "redirect:/";
    }
}
```

}

5. Настройте приложение:

```
java Copy code  
  
// Application.java  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Application {  
    public static void main(String[] args) {  
        SpringApplication.run(Application.class, args);  
    }  
}
```

6. Создайте HTML-шаблоны для отображения данных:

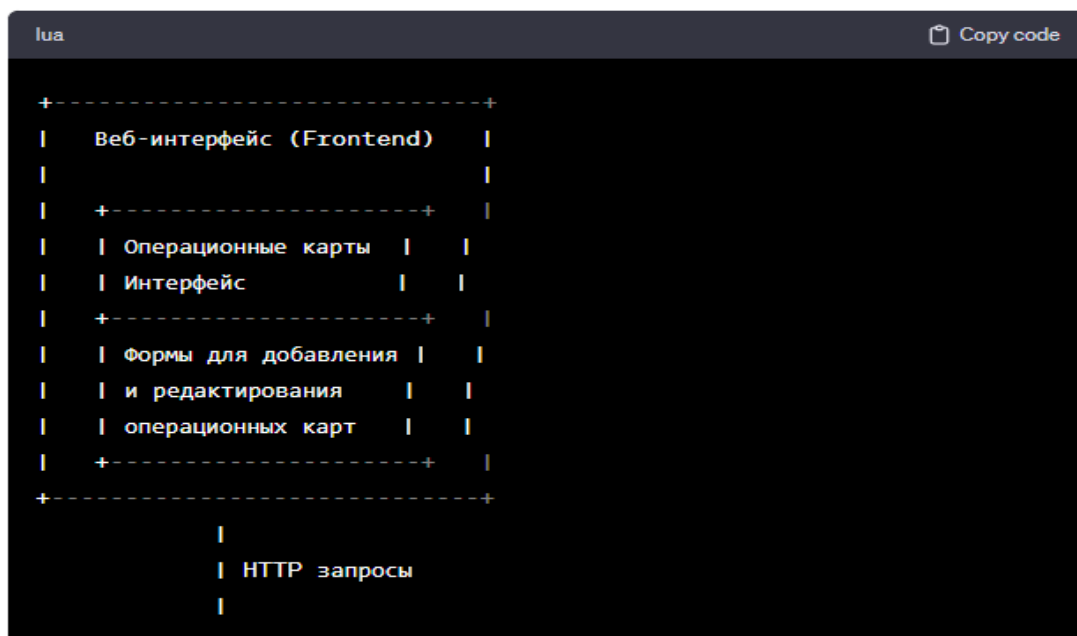
```
html Copy code  
  
<!-- src/main/resources/templates/index.html -->  
<!DOCTYPE html>  
<html>  
<head>  
    <title>Operative Cards</title>  
</head>  
<body>  
    <h1>Operative Cards</h1>  
    <ul>  
        <li th:each="card : ${cards}">  
            <span th:text="${card.title}"></span>  
            <span th:text="${card.description}"></span>  
            <span th:text="${card.status}"></span>  
        </li>  
    </ul>  
    <form th:action="@{/add_card}" th:object="${newCard}" method="post">  
        <label>Title: <input type="text" th:field="*{title}" /></label>  
        <label>Description: <input type="text" th:field="*{description}" /></label>  
        <label>Status: <input type="text" th:field="*{status}" /></label>  
        <button type="submit">Add Card</button>  
    </form>  
</body>  
</html>
```

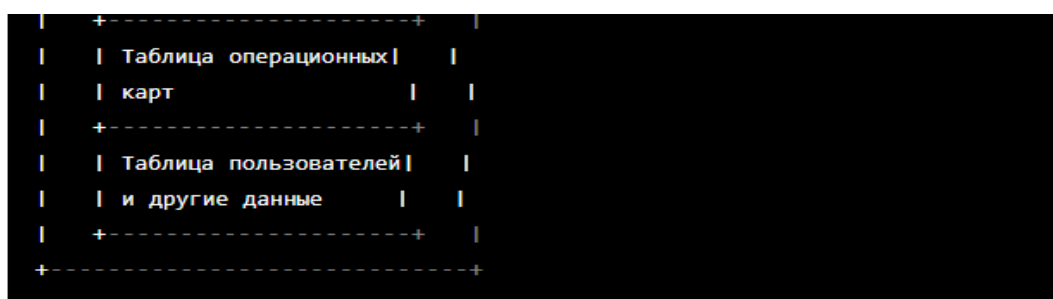
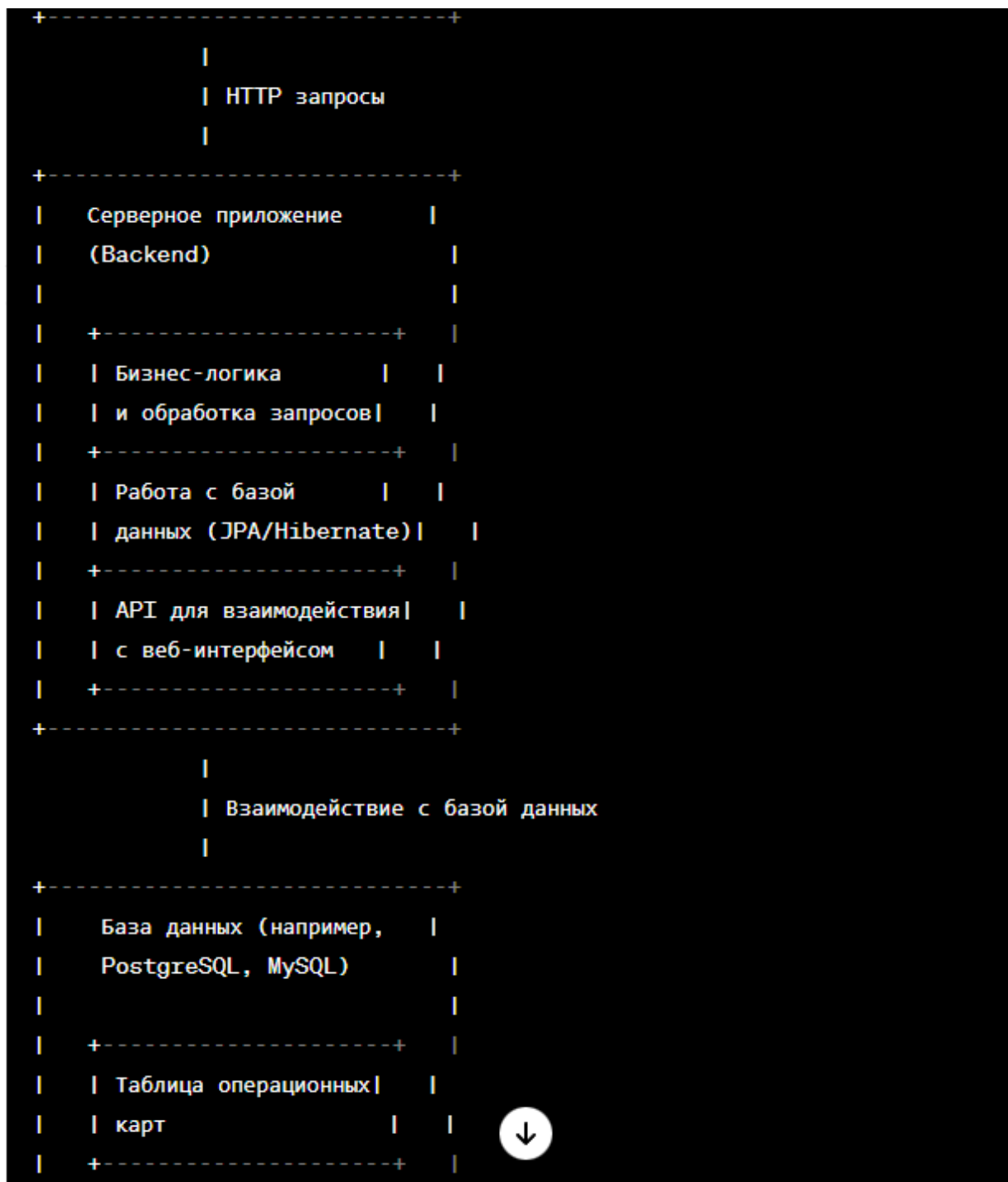


Схема на ascii для проекта "Автоматизированная система управления операционными картами"

ChatGPT

Извините за путаницу, но создание полноценной схемы проекта в текстовом формате ASCII может быть сложным и не всегда передает все аспекты дизайна и структуры. Однако, я могу предоставить простую текстовую диаграмму, чтобы показать основные компоненты системы и их взаимосвязи:





Это простая диаграмма, которая показывает взаимодействие между веб-интерфейсом, серверным приложением и базой данных в автоматизированной системе управления операционными картами. На практике, структура проекта может быть более сложной, в зависимости от требований и архитектурных решений.