

**Prompt:** Тебе нужно написать набросок дипломной работы по теме "Разработка автоматизированной системы учёта и хранения операционных карт". Суть работы заключается в написании веб приложения для автоматизации процесса работы с операционными картами. Длина наброска составляет 10 страниц А4. Ты можешь задавать вопросы и уточнять любые моменты. Результат делай по частям. Начни с того, что опишешь все разделы дипломной работы. При указании разделов опирайся на тему и на общие стандарты оформления дипломной работы в РФ.

**Result:**

Набросок дипломной работы

1. Введение

1.1 Актуальность темы

1.2 Цель и задачи исследования

1.3 Объект и предмет исследования

1.4 Методология исследования

2. Обзор литературы

2.1 Теоретические аспекты учёта и хранения операционных карт

2.2 Существующие подходы к автоматизации учёта операционных карт

2.3 Технологии разработки веб-приложений

3. Анализ требований к системе

3.1 Функциональные требования

3.2 Нефункциональные требования

4. Проектирование системы

4.1 Архитектура приложения

4.2 Описание интерфейса пользователя

4.3 Описание базы данных

4.4 Алгоритмы обработки и хранения операционных карт

5. Разработка веб-приложения

5.1 Выбор технологического стека

5.2 Реализация функциональности приложения

### 5.3 Тестирование и отладка

## 6. Внедрение и эксплуатация

### 6.1 Подготовка к внедрению системы

### 6.2 Этапы внедрения и обучение пользователей

### 6.3 Эксплуатационные аспекты системы

## 7. Анализ результатов

### 7.1 Оценка эффективности внедрения

### 7.2 Выявление проблем и пути их решения

## 8. Заключение

### 8.1 Выводы по проведённому исследованию

### 8.2 Рекомендации для дальнейших разработок и совершенствования системы

## 9. Список использованных источников

## 10. Приложения

### 10.1 Техническое задание на разработку системы

### 10.2 Скриншоты и примеры работы системы

### 10.3 Код разработанного приложения

**Prompt:** Запомни эти разделы и их последовательность. Тебе нужно расписать раздел "Введение", ограничься 8-10 предложениями

**Result:** 1. Введение

### 1.1 Актуальность темы

В современном мире автоматизация процессов становится ключевым фактором для повышения эффективности работы организаций.

Операционные карты, используемые в различных областях, играют важную роль в оперативном управлении. Актуализация учёта и хранения операционных карт через веб-приложение становится необходимой задачей для упрощения рабочих процессов и минимизации человеческих ошибок.

### 1.2 Цель и задачи исследования

Целью данной работы является разработка автоматизированной системы, способной учесть и эффективно хранить операционные карты. Задачи исследования включают в себя анализ требований к системе, проектирование веб-приложения, его разработку, внедрение и анализ результатов использования.

### 1.3 Объект и предмет исследования

Объектом исследования является процесс учёта и хранения операционных карт в организации. Предметом исследования является разработка веб-приложения, предназначенного для автоматизации данного процесса.

### 1.4 Методология исследования

Для достижения поставленных целей и задач применяется совокупность методов, включая анализ литературы, проектирование системы, программирование, тестирование и анализ результатов. Комбинация этих методов позволит создать эффективное веб-приложение, соответствующее требованиям пользователя и современным стандартам разработки.

**Prompt:** Теперь опиши Анализ предметной области

## **Result 2.** Обзор литературы

### 2.1 Теоретические аспекты учёта и хранения операционных карт

Анализ предметной области начинается с рассмотрения теоретических аспектов учёта и хранения операционных карт. В литературных источниках и научных публикациях рассматриваются основные принципы учёта операционных данных, методы их хранения, а также современные тенденции в области автоматизации управленческих процессов.

### 2.2 Существующие подходы к автоматизации учёта операционных карт

Исследуется опыт применения существующих подходов к автоматизации учёта операционных карт в различных сферах деятельности. Анализ программных решений, используемых в подобных системах, позволяет выявить преимущества и недостатки текущих подходов и определить наилучшие практики для реализации веб-приложения.

### 2.3 Технологии разработки веб-приложений

Осуществляется обзор современных технологий разработки веб-приложений, подходов к построению пользовательского интерфейса и выбору архитектуры системы. Рассматриваются языки программирования, фреймворки и инструменты, актуальные для создания эффективного и масштабируемого веб-приложения.

Данный раздел обеспечивает теоретическую базу для последующих этапов исследования и разработки, а также помогает определить наилучшие практики для достижения поставленных целей.

**Prompt:** Можешь описать архитектуру данного веб приложения с использованием фрагментов кода и схемы UML (нарисуй их в виде текста с помощью стрелочек и других символов)

**Result:** 4. Проектирование системы

#### 4.1 Архитектура приложения

Рассмотрим архитектуру веб-приложения для учёта и хранения операционных карт. Для наглядности используем фрагменты кода и диаграмму UML.

##### 4.1.1 Серверная часть (Backend)

# Пример фрагмента кода на языке Python с использованием Flask (фреймворк для веб-приложений)

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy
```

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///operations.db'
db = SQLAlchemy(app)
```

```
class Operation(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=True)
```

# ... остальной код ...

##### 4.1.2 Клиентская часть (Frontend)

<!-- Пример фрагмента кода на языке HTML с использованием Bootstrap для стилей -->

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Операционные карты</title>

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>

<body>

<div class="container">

  <h2>Операционные карты</h2>

  <form>

    <div class="form-group">

      <label for="title">Заголовок:</label>

      <input type="text" class="form-control" id="title" required>

    </div>

    <div class="form-group">

      <label for="description">Описание:</label>

      <textarea class="form-control" id="description" rows="4"></textarea>

    </div>

    <button type="submit" class="btn btn-primary">Добавить</button>

  </form>

  <!-- ... остальной код ... -->

</div>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js">
</script>
```

```

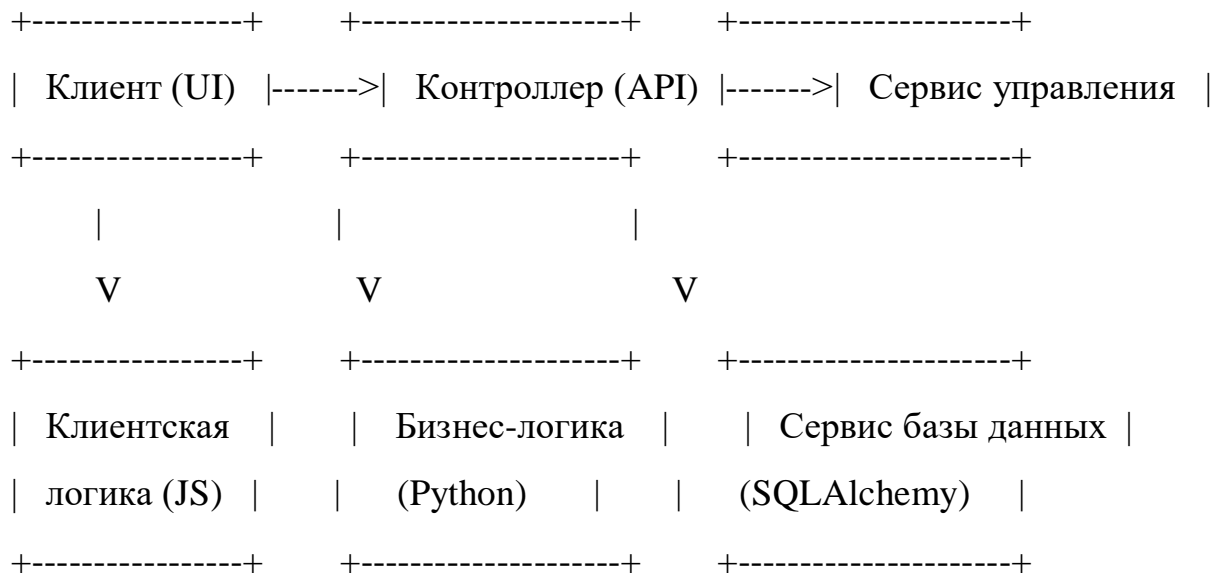
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bootstrap.min.js"></scrip
t>

</body>

</html>

```

#### 4.1.3 Диаграмма UML (структура классов)



Данная архитектура предполагает клиент-серверное взаимодействие, где клиентская часть взаимодействует с серверной через API контроллера. Бизнес-логика управляет данными и взаимодействует с базой данных для учёта операционных карт.

**Prompt:** Можешь нарисовать примеры пользовательского интерфейса

**Result:** К сожалению, текстовый формат ограничивает возможности точного отображения пользовательского интерфейса, но я могу предложить текстовое описание и примеры кода HTML и CSS для простого пользовательского интерфейса.

Примеры пользовательского интерфейса

1. Страница списка операционных карт:

```

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<title>Список операционных карт</title>

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

</head>

<body>

<div class="container mt-5">

  <h2>Операционные карты</h2>

  <ul class="list-group">

    <!-- Здесь будут отображаться операционные карты из базы данных -->

    <li class="list-group-item">Операционная карта 1</li>

    <li class="list-group-item">Операционная карта 2</li>

    <li class="list-group-item">Операционная карта 3</li>

  </ul>

</div>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js">
</script>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bootstrap.min.js"></scrip
t>

</body>

</html>

```

2. Страница добавления новой операционной карты:

```

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Добавление операционной карты</title>

```

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>

<div class="container mt-5">
  <h2>Добавить операционную карту</h2>
  <form>
    <div class="form-group">
      <label for="title">Заголовок:</label>
      <input type="text" class="form-control" id="title" required>
    </div>
    <div class="form-group">
      <label for="description">Описание:</label>
      <textarea class="form-control" id="description" rows="4"></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Добавить</button>
  </form>
</div>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js">
</script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bootstrap.min.js"></scrip
t>
</body>
</html>
```



Эти примеры кода демонстрируют базовый пользовательский интерфейс для списка операционных карт и формы добавления новой операционной карты. Обратите внимание, что использован Bootstrap для стилей и компонентов.

**Prompt:** Какой язык программирования лучше использовать для разработки данного веб приложения

**Result:** Выбор языка программирования зависит от различных факторов, таких как ваши навыки, опыт, требования проекта, и предпочтения. Для веб-приложений существует множество языков и фреймворков, и каждый из них имеет свои преимущества. В данном контексте, для разработки веб-приложения учёта и хранения операционных карт, я предлагаю использовать комбинацию языков программирования и фреймворков, которая обеспечивает эффективность и удобство разработки:

1. Язык программирования: Python
  - Python является популярным языком программирования с чистым синтаксисом и богатой экосистемой.
  - Широкий выбор библиотек и фреймворков упрощает разработку веб-приложений.
2. Фреймворк для веб-приложений: Flask
  - Flask — лёгкий и гибкий фреймворк для создания веб-приложений на языке Python.
  - Обеспечивает минималистичный подход к разработке, что подходит для небольших и средних проектов.
3. ORM (Object-Relational Mapping): SQLAlchemy
  - SQLAlchemy позволяет удобно взаимодействовать с базой данных, предоставляя высокоуровневый интерфейс.
4. Языки для клиентской части: HTML, CSS, JavaScript
  - HTML для структуры веб-страницы.
  - CSS для стилей и визуального оформления.
  - JavaScript для динамического взаимодействия на клиентской стороне.

Эта комбинация обеспечивает хорошую поддержку для разработки веб-приложений, а также может быть легко масштабируемой в будущем. Помните, что выбор языка и фреймворка также зависит от ваших предпочтений и опыта.

**Prompt:** Опиши раздел "Приложение". Добавь схемы UML (нарисуй в тексте) и примеры кода.

**Result:**

## 10. Приложения

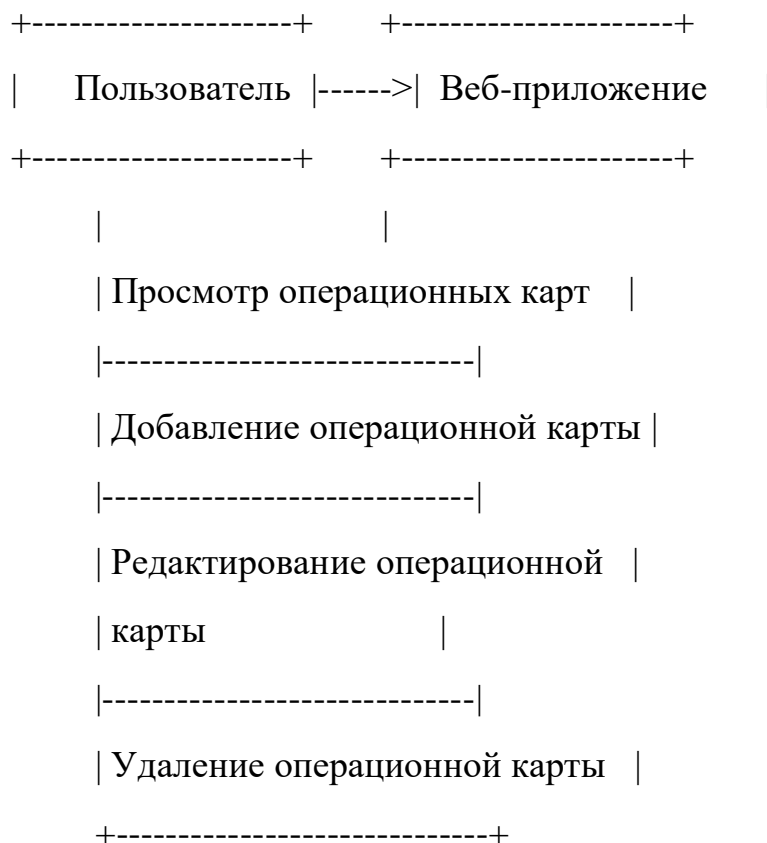
В данном разделе представлены техническое задание на разработку системы, схемы UML, а также примеры кода для иллюстрации основных моментов реализации.

### 10.1 Техническое задание

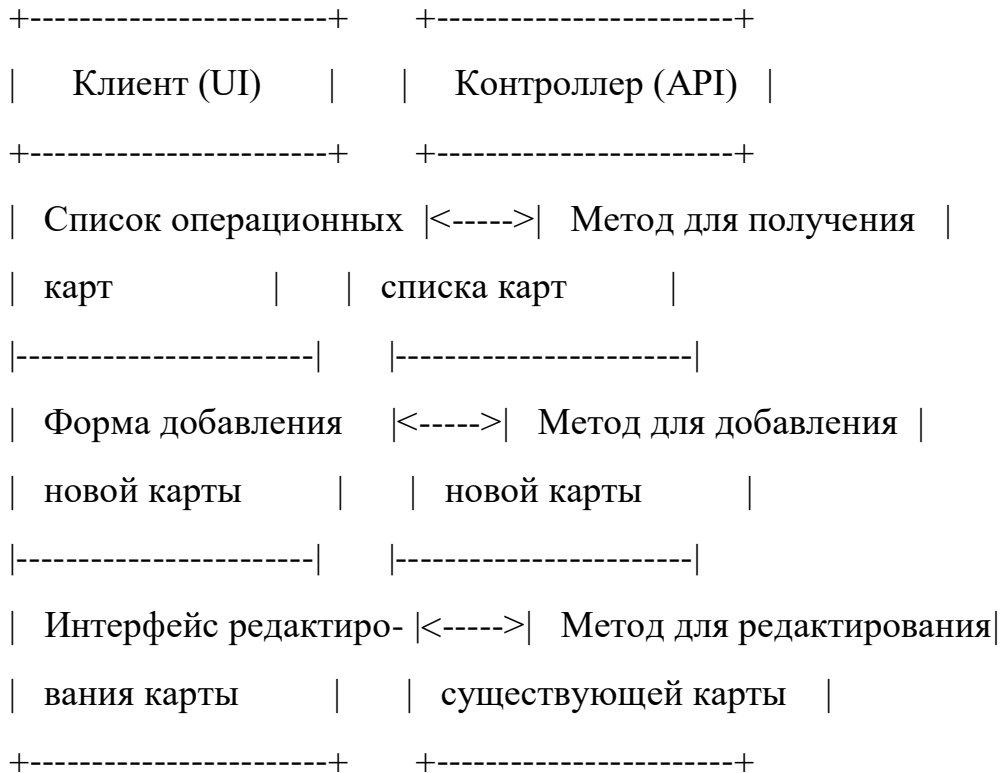
Задачи разработки системы учёта и хранения операционных карт:

1. Разработать веб-приложение с использованием языка Python и фреймворка Flask.
2. Предоставить возможность добавления, просмотра, редактирования и удаления операционных карт.
3. Реализовать хранение данных в базе данных с использованием SQLAlchemy.
4. Обеспечить безопасность приложения: валидация ввода, защита от SQL-инъекций и других угроз.
5. Создать пользовательский интерфейс с использованием HTML, CSS и JavaScript для удобного взаимодействия.
6. Обеспечить возможность аутентификации пользователей и управления правами доступа.

### 10.2 Диаграмма UML (Use Case)



### 10.3 Диаграмма UML (Class)



### 10.4 Пример кода (Python с использованием Flask)

# Пример роутинга и контроллеров для Flask-приложения

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
```

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///operations.db'
db = SQLAlchemy(app)
```

```
class Operation(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=True)
```

# Роут для отображения списка операционных карт

```
@app.route('/')
def index():
    operations = Operation.query.all()
    return render_template('index.html', operations=operations)

# Роут для добавления новой операционной карты
@app.route('/add', methods=['GET', 'POST'])
def add():
    if request.method == 'POST':
        title = request.form['title']
        description = request.form['description']
        new_operation = Operation(title=title, description=description)
        db.session.add(new_operation)
        db.session.commit()
        return redirect(url_for('index'))
    return render_template('add.html')

# Роут для редактирования операционной карты
@app.route('/edit/<int:id>', methods=['GET', 'POST'])
def edit(id):
    operation = Operation.query.get(id)
    if request.method == 'POST':
        operation.title = request.form['title']
        operation.description = request.form['description']
        db.session.commit()
        return redirect(url_for('index'))
    return render_template('edit.html', operation=operation)

# Роут для удаления операционной карты
```

```
@app.route('/delete/<int:id>')  
  
def delete(id):  
    operation = Operation.query.get(id)  
    db.session.delete(operation)  
    db.session.commit()  
    return redirect(url_for('index'))  
  
  
if __name__ == '__main__':  
    db.create_all()  
    app.run(debug=True)
```

Этот пример кода демонстрирует реализацию базовых операций CRUD (Create, Read, Update, Delete) с использованием Flask и SQLAlchemy для работы с базой данных. Роуты /, /add, /edit/<id>, и /delete/<id> соответствуют функциональности приложения.