УНИВЕРСИТЕТ «УРБАН»

**ДИПЛОМНАЯ РАБОТА**

**на тему**:  
Разработка веб-приложения для работы с торговыми точками и базой дебиторов

**Автор**: Перепелица Максим

г. Минеральные Воды 2024 г.

Оглавление

[1. Введение 3](#_Toc183088815)

[1.1 Обоснование выбора темы 3](#_Toc183088816)

[1.2 Определение цели и задач работы 3](#_Toc183088817)

[2. Основные понятия и определения 4](#_Toc183088818)

[3. Методы и подходы к разработке 4](#_Toc183088819)

[3.1 Выбор подходящего фреймворка для разработки 4](#_Toc183088820)

[3.2 Архитектура веб-приложения 4](#_Toc183088821)

[3.3 Обеспечение безопасности 4](#_Toc183088822)

[Защита данных пользователей 4](#_Toc183088823)

[4. Обзор популярных инструментов для разработки веб-приложений на Python 7](#_Toc183088824)

[4.1 Джанго 7](#_Toc183088825)

[4.2 FastAPI 7](#_Toc183088826)

[4.3 Колба 7](#_Toc183088827)

[5. Проектирование приложения 7](#_Toc183088828)

[5.1 Планирование и анализ требований 7](#_Toc183088829)

[5.2 Основные функциональные и технические требования 8](#_Toc183088830)

[6. Разработка приложения в соответствии с документацией 8](#_Toc183088831)

[7. Анализ результатов 8](#_Toc183088832)

[8. Заключение 8](#_Toc183088833)

[8.1 Обзор выполненной работы 8](#_Toc183088834)

[8.2 Возможности для дальнейшего развития 8](#_Toc183088835)

# 1. Введение

## 1.1 Обоснование выбора темы

Современные бизнес-процессы требуют оперативного доступа к информации и эффективного управления финансовыми рисками. В торговле это особенно важно при взаимодействии с клиентами, где своевременная проверка статуса торговых точек и управление дебиторской задолженностью помогают минимизировать убытки.

Многие предприятия малого и среднего бизнеса сталкиваются с проблемой отсутствия специализированных инструментов, которые могли бы интегрировать функции мониторинга статуса торговых точек и ведения базы данных в одном приложении.

Для решения этой проблемы был разработан проект **«Дебитор-Трекер»**, веб-приложение для торговых представителей, которое помогает:

* быстро проверять статус торговых точек в базе дебиторов;
* добавлять новые точки в систему;
* получать доступ к актуальной информации о задолженностях.

Таким образом, разработка веб-приложения «Дебитор-Трекер» направлена на повышение эффективности работы торговых представителей и минимизацию рисков, связанных с просроченной дебиторской задолженностью.

## 1.2 Определение цели и задач работы

**Цель работы** — разработка веб-приложения, которое позволит торговым представителям легко и быстро работать с базой дебиторов и добавлять новые торговые точки.

**Задачи:**

* Создать веб-интерфейс с интуитивно понятным дизайном для отображения информации о статусе торговых точек.
* Реализовать функционал добавления новых торговых точек в систему.
* Обеспечить поиск и фильтрацию данных для удобства работы.
* Разработать безопасную систему авторизации и регистрации пользователей.
* Гарантировать возможность масштабирования приложения для обработки больших объёмов данных.

## 2. Основные понятия и определения

* **Дебитор**: клиент, имеющий задолженность перед компанией.
* **Фронтенд**: часть приложения, отвечающая за взаимодействие с пользователем (интерфейс).
* **Бэкенд**: серверная часть приложения, реализующая бизнес-логику.
* **Фреймворк**: программная платформа, предоставляющая инструменты для разработки.

# 3. Методы и подходы к разработке

## 3.1 Выбор подходящего фреймворка для разработки

Для реализации проекта был выбран фреймворк **Django**, который обладает встроенными инструментами для работы с базами данных, обеспечивает безопасность и поддерживает масштабируемость.

## 3.2 Архитектура веб-приложения

Архитектура приложения основана на шаблоне MVT (Model-View-Template):

* **Модели (Models)**: отвечают за управление данными и связь с базой данных.
* **Шаблоны (Templates)**: представляют визуальную часть приложения.
* **Контроллеры (Views)**: реализуют логику взаимодействия между пользователем и данными.

## 3.3 Обеспечение безопасности

### Защита данных пользователей

Для обеспечения безопасности данных пользователей в проекте реализованы следующие механизмы:

#### 1. Аутентификация и авторизация

В Django встроена мощная система аутентификации и авторизации, которая позволяет:

* Управлять пользователями (создание, изменение, удаление).
* Ограничивать доступ к различным страницам или функциональности приложения.
* Поддерживать авторизацию на основе сессий и токенов.

**Пример реализации аутентификации:**

В settings.py добавлена стандартная система аутентификации Django:

python

# settings.py

AUTHENTICATION\_BACKENDS = [

'django.contrib.auth.backends.ModelBackend', # Основная система аутентификации

]

**Код формы входа в систему (views.py):**

python

Копировать код

from django.contrib.auth import authenticate, login

from django.shortcuts import render, redirect

from django.contrib import messages

def user\_login(request):

if request.method == 'POST':

username = request.POST['username']

password = request.POST['password']

user = authenticate(request, username=username, password=password) # Проверка пользователя

if user is not None:

login(request, user) # Авторизация

return redirect('dashboard') # Перенаправление на главную страницу после входа

else:

messages.error(request, "Неправильное имя пользователя или пароль")

return render(request, 'login.html')

**Шаблон HTML для авторизации (login.html):**

html

<form method="post">

{% csrf\_token %}

<label for="username">Имя пользователя:</label>

<input type="text" name="username" id="username" required>

<label for="password">Пароль:</label>

<input type="password" name="password" id="password" required>

<button type="submit">Войти</button>

</form>

#### 2. Шифрование данных

Пароли пользователей автоматически хэшируются с использованием встроенной функции Django make\_password. Это гарантирует, что даже в случае утечки базы данных пароли останутся безопасными.

**Создание нового пользователя (пример в Django Admin):**

python

from django.contrib.auth.models import User

# Создание нового пользователя

new\_user = User.objects.create\_user(username='example\_user', password='secure\_password123')

new\_user.save()

#### 3. Защита от SQL-инъекций

Django ORM (Object-Relational Mapping) предотвращает SQL-инъекции, так как запросы к базе данных автоматически экранируются.

**Пример безопасного запроса:**

python

from myapp.models import TradePoint

# Безопасный фильтр по имени точки

trade\_point = TradePoint.objects.filter(name='Точка №1')

Этот подход предотвращает внедрение вредоносного SQL-кода через пользовательский ввод.

#### 4. Использование CSRF-токенов для защиты форм

Django автоматически добавляет CSRF-токены ко всем POST-запросам, чтобы предотвратить атаки типа Cross-Site Request Forgery (CSRF).

**Пример использования CSRF в шаблоне:**

html

<form method="post">

{% csrf\_token %}

<input type="text" name="example\_field" required>

<button type="submit">Отправить</button>

</form>

#### 5. Дополнительные меры безопасности

* Включение HTTPS для шифрования данных между клиентом и сервером.
* Регулярное обновление Django и используемых библиотек для устранения известных уязвимостей.

Эти меры вместе обеспечивают высокий уровень защиты данных пользователей в веб-приложении.

# 4. Обзор популярных инструментов для разработки веб-приложений на Python

## 4.1 Джанго

Основной фреймворк разработки, поддерживающий:

* ORM для работы с базой данных.
* Аутентификацию пользователей.
* Инструменты для администрирования данных.

## 4.2 FastAPI

Асинхронный фреймворк для создания API. Обеспечивает высокую производительность.

## 4.3 Колба

Минималистичный фреймворк, предоставляющий гибкость в разработке.

# 5. Проектирование приложения

## 5.1 Планирование и анализ требований

Составлены схемы взаимодействия фронтенда, бэкенда и базы данных. Приложение содержит три основных модуля:

1. Управление базой дебиторов.
2. Поиск и фильтрация точек.
3. Добавление новых торговых точек.

## 5.2 Основные функциональные и технические требования

1. База данных SQLite: используется для хранения информации о торговых точках и пользователях. SQLite обеспечивает простоту настройки и подходит для небольших и средних приложений, что делает её идеальным выбором для начального этапа разработки.
2. Система поиска и фильтрации данных: реализована для удобного доступа к информации о торговых точках.
3. Защищённая авторизация: реализован

# 6. Разработка приложения в соответствии с документацией

Разработка включала этапы:

1. Создание базовых моделей данных.
2. Реализация интерфейсов с использованием Bootstrap.
3. Разработка API для взаимодействия между фронтендом и бэкендом.
4. Тестирование функционала.

# 7. Анализ результатов

Веб-приложение позволяет:

* Торговым представителям быстро проверять статусы точек.
* Эффективно управлять базой дебиторов.
* Избегать финансовых рисков, связанных с просроченными задолженностями.

# 8. Заключение

## 8.1 Обзор выполненной работы

Разработано удобное веб-приложение для торговых представителей, упрощающее управление базой дебиторов.

## 8.2 Возможности для дальнейшего развития

* Добавление функционала автоматической интеграции с CRM-системами.
* Расширение базы данных для работы с большими объёмами информации.