МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра математической теории упругости и биомеханики

Разработка приложения по учету внерабочих активностей

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 442 группы

направление 09.03.03 – Прикладная информатика

механико-математического факультета

Потапова Мирослава Андреевича

Научный руководитель
доцент, к.ю.н. Р.В. Амелин

Зав. кафедрой
зав. кафедрой, д.ф.-м.н., профессор Л.Ю. Коссович

Саратов 2025

Введение. Веб-приложение «Event Tracker» предлагает решение для оптимизации процесса управления событиями и коммуникации. Идея проекта – разработка веб-приложения, которое позволяет эффективно управлять событиями, настраивать уведомления и получать их через Telegram. Предлагаемый продукт будет способствовать повышению эффективности планирования и коммуникации, отвечая на растущую потребность в автоматизации процессов управления событиями.

Актуальность проекта обосновывается растущим спросом на системы управления событиями в условиях цифровизации бизнес-процессов. Исследования рынка показали, что существующие решения часто не обеспечивают необходимой гибкости в учете и не предлагают интеграцию с мессенджером. Анализ конкурентных продуктов позволил сформулировать требования к разрабатываемому приложению и определить его конкурентные преимущества.

Основная цель проекта – разработка системы, позволяющей пользователям эффективно управлять событиями и получать своевременные уведомления через Telegram. Целью данной бакалаврской работы является создание серверной части (бэкенда) для проекта, удовлетворяющей современным требованиям разработки веб-приложений.

Целью данной бакалаврской работы является разработка бэкенда приложения. Она представляет собой ключевую составляющую создания современных веб-приложений, обеспечивающую обработку данных, выполнение бизнес-логики и взаимодействие с базами данных. В условиях развития микросервисной архитектуры и контейнеризации проектирование и реализация серверной части становятся критически важными задачами. Все это определяет актуальность темы бакалаврской работы.

К современным приложениям предъявляются высокие требования к безопасности, масштабируемости и производительности.

За счет использования микросервисной архитектуры, Docker и современных фреймворков обеспечивается гибкость системы, позволяющая оперативно вносить изменения и расширять функциональность.

Для достижения целей проекта были поставлены следующие **задачи**:

– провести анализ существующих решений и сформулировать требования к приложению;

– разработать архитектуру серверной части с использованием микросервисов;

– реализовать систему уведомлений с интеграцией Telegram API;

– внедрить механизмы защиты от SQL-инъекций;

– настроить контейнеризацию и оркестрацию с помощью Docker Compose;

– обеспечить высокую производительность через двухуровневую архитектуру (Nginx + Gunicorn);

– запустить и протестировать приложение в production-окружении.

Структура и объем работы. Работа состоит из введения, трех разделов, двух приложений и содержит 49 страниц. Список использованных источников включает 25 наименований.

Раздел 1. Анализ предметной области и разработка требований.

Раздел 2. Проектирование информационной системы приложения.

Раздел 3. Реализация в коде.

Основное содержание работы. Работа посвящена разработке приложения «Event Tracker».

Во введении обосновывается актуальность проекта, связанного с разработкой веб-приложения для учета внерабочих активностей сотрудников. Основная цель работы — создание серверной части (бэкенда) приложения, отвечающего современным требованиям безопасности, производительности и масштабируемости. Поставлены задачи, включающие анализ предметной области,

проектирование архитектуры, реализацию функционала и интеграцию с внешними сервисами, такими как Telegram API.

В первом разделе исследуется важность учета внерабочих активностей в IT-сфере, выделяя их роль в профессиональном развитии сотрудников и укреплении корпоративной культуры.

Внерабочими активностями называют различные виды профессионального развития и взаимодействия, которые сотрудники осуществляют вне рамок своей основной рабочей деятельности. К таким активностям относят:

- выступления на конференциях и форумах;
- участие в хакатонах и конкурсах;
- организация и проведение семинаров и тренингов;
- наставничество и помощь новым сотрудникам;
- публикации научных работ и аналитических обзоров.

Эти активности способствуют профессиональному росту и развитию личного бренда сотрудника, повышая его ценность как специалиста.

Современный подход к учёту профессиональной активности сотрудников характеризуется рядом недостатков:

- информация разбросана между различными источниками хранения данных (электронные письма, таблицы, мессенджеры);
- процесс сбора данных трудоемкий и требует значительных усилий со стороны менеджеров и специалистов по персоналу;
- отсутствуют удобные инструменты для формирования отчетов и анализа данных;
- существует высокий риск потери важных сведений при смене сотрудников или переходе их на новую должность.

Веб-приложение разрабатывалось для удовлетворения потребностей руководителей команд и работодателей в IT-индустрии. Оно решает проблему оценки вклада каждого сотрудника в профессиональное сообщество и позволяет точно определить степень его участия в коллективных инициативах.

Разработаны функциональные и нефункциональные требования к приложению, включая поддержку уведомлений через Telegram, формирование отчетов и обеспечение безопасности данных.

Проект уникален тем, что оно сочетает функциональность всех перечисленных сервисов и дополняет их новыми возможностями:

– централизация данных: Все сведения о сотрудниках и их участии хранятся в одном месте, позволяя быстро получать нужную информацию;

– автоматическая отчётность: Система генерирует отчёты, позволяющие руководителям видеть общую картину активности сотрудников и эффективность отдельных направлений;

– интерфейсы интеграции: Пользователи могут подключать сторонние сервисы, интегрироваться с корпоративными порталами и CRM-системами;

– детализированная статистика: Показатели включают число выступлений, проведённых курсов, публикаций и других видов активности, помогающих принять обоснованные управленческие решения;

– отправка уведомлений и сбор фидбека: Функция отправки уведомлений о мероприятиях и сборе отзывов о прошедших событиях делает приложение незаменимым помощником для всех заинтересованных лиц.

Во втором разделе описана архитектура приложения, включая:

Базу данных: Использована PostgreSQL, обоснован выбор её надежности, производительности и поддержки сложных запросов. На рисунке 1 приведена схема таблиц (пользователи, события, уведомления и др.) и их взаимосвязей.

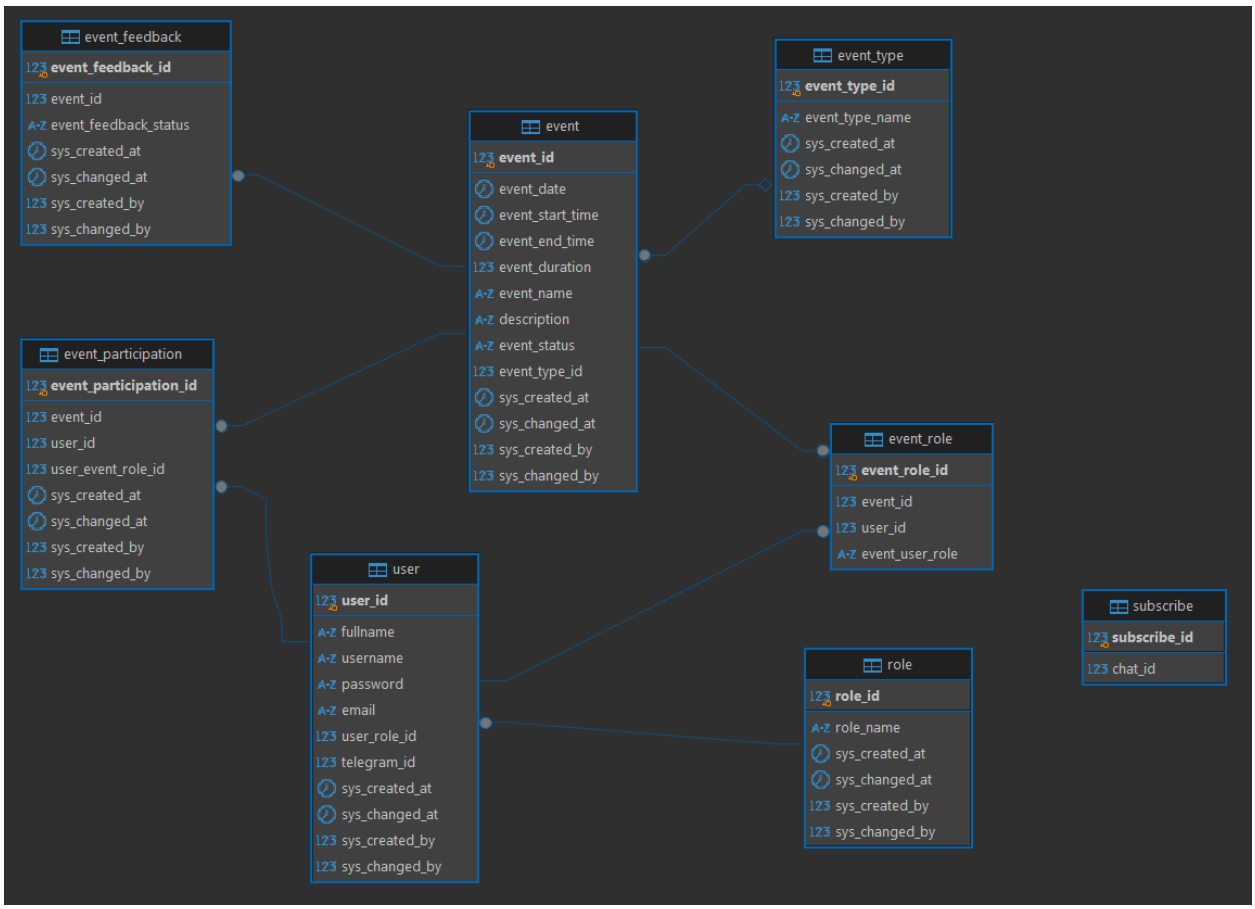


Рисунок 1 – Схема базы данных приложения

Серверную часть: Реализована на Flask с модульной структурой (Blueprint), что обеспечивает гибкость и масштабируемость. На рисунке 2 видны ключевые компоненты:

– app.py — основной модуль, обрабатывает запросы, управляет аутентификацией и взаимодействием между частями системы.

– check_events.py — проверяет события, может работать по расписанию через Airflow.

– tg_notif.py — отправляет уведомления через Telegram API.

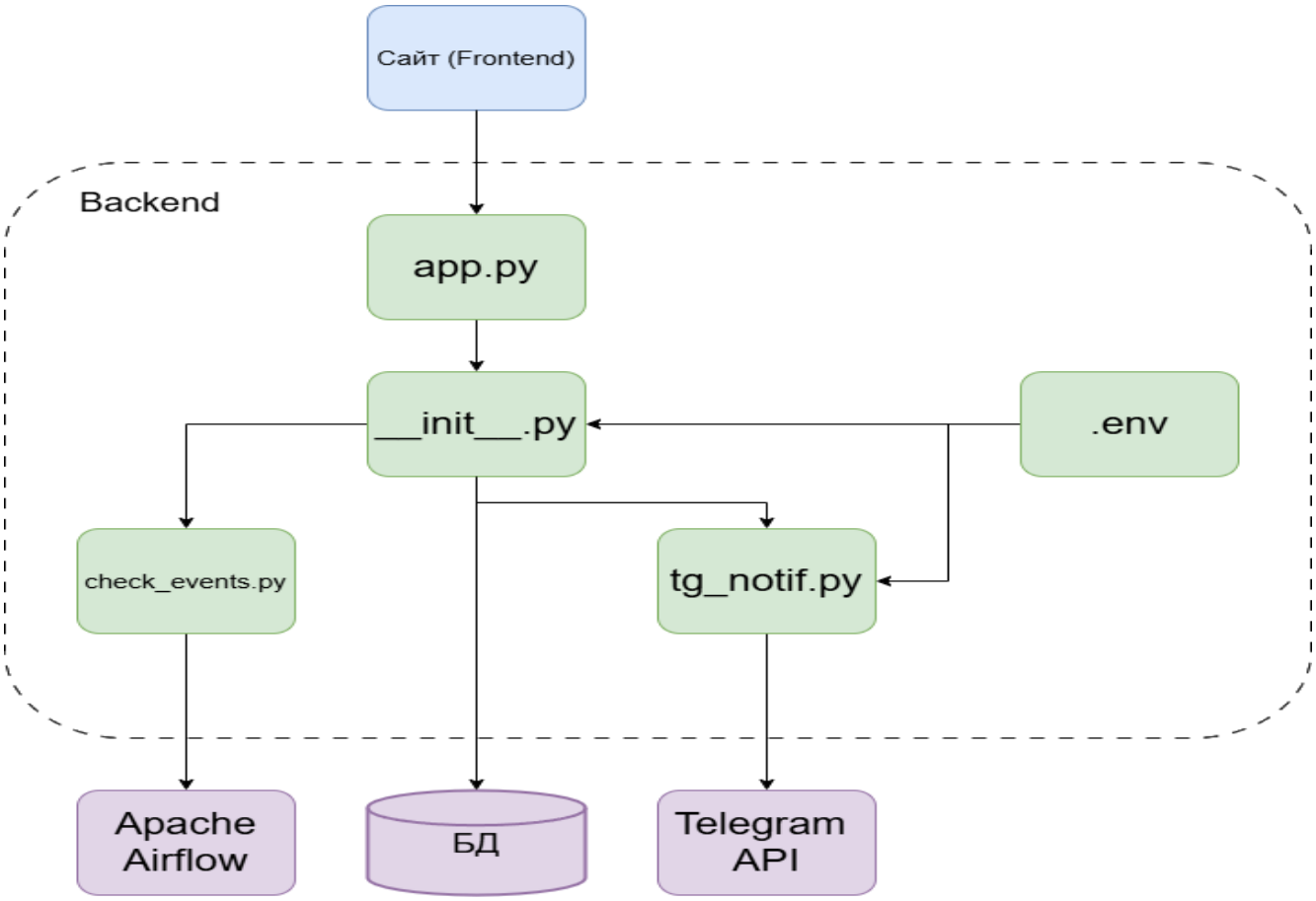
– .env — хранит конфигурационные переменные (ключи, пароли, настройки).

– Frontend — сайт для пользователей.

– БД — база данных для хранения информации.

– Apache Airflow — автоматизация задач.

– Telegram API — внешний сервис для уведомлений.



Риснок 2 – Архитектура приложения на уровне компонентов

Фронтенд: Разработан на HTML/CSS/JavaScript с использованием Bootstrap для адаптивного дизайна. Взаимодействие с бэкендом осуществляется через HTTP-запросы.

В третьем разделе подробно рассматривается практическая реализация проекта — система трекинга событий с уведомлениями и автоматизацией. В качестве основы выбран фреймворк Flask, который идеально подходит для небольших и средних проектов благодаря своей минималистичной архитектуре и возможности гибкого расширения. В основе Flask лежат два ключевых компонента:

Werkzeug — библиотека, реализующая стандарт WSGI (Web Server Gateway Interface), который обеспечивает взаимодействие между веб-сервером и Python-приложением. Werkzeug отвечает за низкоуровневую обработку HTTP-запросов и

формирование ответов, что позволяет гибко управлять маршрутизацией и обработкой данных.

Jinja2 — современная система шаблонов, интегрируемая в HTML-документы для динамической генерации контента. Jinja2 компилирует шаблоны в оптимизированный Python-код, что ускоряет рендеринг страниц и позволяет удобно разделять логику и представление.

Файл .env загружается при запуске приложения и содержит все необходимые переменные окружения: параметры подключения к базе данных PostgreSQL, токены для интеграции с Telegram API, секретные ключи и другие настройки. Это обеспечивает централизованное и безопасное хранение конфигурации, избавляя от необходимости жестко прописывать параметры в исходном коде.

Модуль check_events.py реализует автоматическую проверку событий. Его задача — анализировать предстоящие события и инициировать необходимые действия, например, отправку напоминаний. Для регулярного запуска этого модуля используется Apache Airflow, который позволяет настраивать расписание и автоматизировать выполнение задач.

Модуль tg_notif.py отвечает за интеграцию с Telegram API. Он реализует логику отправки уведомлений пользователям о предстоящих событиях или изменениях в расписании. Для этого используются токены и параметры, хранящиеся в .env.

Модуль app.py является центральным элементом backend-приложения. В нем происходит инициализация Flask-приложения, настройка маршрутов, подключение всех вспомогательных модулей, а также реализация аутентификации пользователей. Через app.py осуществляется взаимодействие между frontend (веб-сайтом), базой данных и внешними сервисами (например, Telegram).

Frontend реализован как отдельный веб-сайт, который взаимодействует с backend через API-вызовы. Пользовательский интерфейс позволяет просматривать события, регистрироваться на них, а также получать уведомления.

База данных PostgreSQL используется для хранения информации о событиях, пользователях и их участии. Взаимодействие с БД осуществляется через соответствующие модули backend-приложения.

Apache Airflow интегрирован для автоматизации и планирования задач, таких как регулярная проверка событий и отправка уведомлений.

Telegram API используется для отправки мгновенных уведомлений пользователям, что повышает оперативность информирования.

В разделе также подробно описывается взаимодействие между модулями: как пользователь регистрируется через формы frontend, как данные передаются в backend и сохраняются в базе данных, как запускаются автоматические задачи и отправляются уведомления. Обоснован выбор серверной платформы и инструментов, исходя из требований к быстрому развертыванию, поддержке Python и интеграции с внешними сервисами.

Заключение. В ходе выполнения бакалаврской работы был проведен анализ потребностей в системах учета внерабочих активностей сотрудников, изучены существующие решения на рынке, сформулированы требования к веб-приложению «Event Tracker» с интеграцией Telegram-уведомлений. Работа выполнена в рамках реального проекта, что позволило учесть практические аспекты внедрения и использования системы. Разработанное приложение направлено на автоматизацию учета участия сотрудников в профессиональных мероприятиях и повышение эффективности коммуникации через систему уведомлений.

Проект соответствует современным стандартам веб-разработки, что обеспечивается использованием микросервисной архитектуры, контейнеризации и современных практик DevOps. Особое внимание было уделено безопасности системы: реализована защита от SQL-инъекций через параметризованные запросы, внедрена система аутентификации и авторизации, обеспечено шифрование передаваемых данных. Использование Docker и Docker Compose позволило создать надежную и масштабируемую инфраструктуру, а

двухуровневая архитектура с Nginx и Gunicorn обеспечила высокую производительность и отказоустойчивость.

Интеграция с Telegram API обеспечила удобный канал доставки уведомлений, а асинхронное программирование оптимизировало обработку запросов. Гибкая архитектура приложения позволяет быстро адаптироваться к изменениям требований и интегрироваться с новыми сервисами.

Таким образом, цель бакалаврской работы достигнута: разработана и внедрена система учета внерабочих активностей, отвечающая современным требованиям к безопасности, производительности и масштабируемости. Все поставленные задачи выполнены, а предложенные направления развития обеспечивают потенциал для дальнейшего совершенствования системы.