ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук НИУ ВШЭ-Нижний Новгород

Образовательная программа «Программная инженерия»

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент департамента  программной инженерии  факультета компьютерных наук  канд. техн. наук | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | |  | | --- | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | |

**Пояснительная записка**

**по курсовой работе**

на тему «Разработка информационной системы для регистрации на мероприятия НИУ ВШЭ – Нижний Новгород»

по направлению подготовки бакалавров 09.03.04 «Программная инженерия»

Исполнитель:   
студенты группы 20ПИ-1

Лапшина Юлия Алексеевна

(в составе группы также Титова Н.Д.)

Руководитель:

декан факультета информатики,

математики и компьютерных наук

Асеева Наталья Владимировна

Нижний Новгород 2023

**Аннотация.**  Данная работа выполнена в рамках группового курсового прикладного проекта, ставящего своей целью разработку регистрационной системы для мероприятий факультета компьютерных наук нижегородского кампуса НИУ ВШЭ. В рамках данной пояснительной записки подробно рассматривается реализация серверной части продукта, включая обоснованный выбор средств разработки, структуру базы данных и алгоритмы их обработки, а также описание работы панели администрирования системы.

**Ключевые слова**: регистрационная система

Оглавление

[1. Введение 4](#_Toc134581202)

[2. Средства и технологии разработки 5](#_Toc134581203)

[2.1 Постановка частного технического задания 5](#_Toc134581204)

[2.2 Выбор фреймворка 6](#_Toc134581205)

[2.3 Выбор СУБД 7](#_Toc134581206)

[3. Описание программного продукта 8](#_Toc134581207)

[3.1 Общая файловая структура 8](#_Toc134581208)

[3.2 Структура базы данных 8](#_Toc134581209)

[3.3 Запросы и передача данных 10](#_Toc134581210)

[3.4 Панель администрирования 11](#_Toc134581211)

[4. Заключение 12](#_Toc134581212)

[Источники 13](#_Toc134581213)

# 1. Введение

Групповая курсовая работа, частью которой является данная пояснительная записка, заключается в разработке информационной системы, решающей задачу электронной регистрации участников на мероприятия, организуемые университетом. В общей части текста работы подробно описаны такие начальные шаги реализации, как обоснование актуальности темы, сбор пользовательских требований и постановка на их основе главных целей работы, анализ рынка продуктов со схожей тематикой. Также именно там описана заключительная стадия разработки – тестирование цельной системы, проверка её на предмет соответствия изначальным критериям, анализ проделанной работы и перспективы расширения.

Что касается этой индивидуальной работы, она рассматривает подробности реализации серверной и алгоритмической части системы (бэкенд, backend). В ней будет конкретизировано обрисованное ранее техническое задание, обоснован выбор технологий и инструментов для разработки и описана основная структура кода и программных компонентов.

# 2. Средства и технологии разработки

## 2.1 Постановка частного технического задания

Сформулированное на уровне пользовательских требований техническое задание заключалось в том, что система должна предоставлять функционал для просмотра списка мероприятий и регистрации на конкретные из них для сторонних участников, а также – для управления данным списком и для просмотра информации о зарегистрировавшихся со стороны администраторов системы. Для конкретизации критериев оценки это было уточнено до уровня функциональных требований, которые образовали следующий список.

* Система должна быть реализована в виде браузерных веб-страниц с опорой на сервер базы данных.
* Открывающаяся первой веб-страница системы должна содержать форму регистрации, которую предлагается заполнить пользователю, указанием имени, места обучения и контактной информации.
* Первая страница также содержит список событий, доступных для регистрации – для каждого конкретного из них доступны для выбора любые дни, входящие в мероприятие.
* Выбрав интересующие его дни и нажав на кнопку «Зарегистрироваться», пользователь должен увидеть вторую веб-страницу, на которой представлено расписание на выбранные дни.
* Пункты расписания разделены на категории по времени, и пользователь не может выбрать больше одного пункта в категории, при том что он обязан сделать выбор в каждой из доступных категорий.
* Пункты расписания, помимо прочей информации о них, имеют количество доступных мест. На мероприятие с ограниченными местами это количество должно автоматически увеличиваться или уменьшаться в зависимости от появления и удаления регистраций. Зарегистрироваться на мероприятие, на которое не осталось мест, должно быть невозможно.
* После выбора во всех категориях и нажатия на кнопку «Завершить регистрацию» пользователь должен увидеть третью веб-страницу с подтверждением успешной отправки данных, а также расписание выбранных им мероприятий.
* На третьей странице должна присутствовать возможность сохранить отображаемый список выбранных мероприятий как самостоятельный файл.
* В процессе регистрации в базе данных должна сохраняться информация об участнике, а также создаваться записи о его регистрациях на конкретные мероприятия, включая упомянутый выше механизм подсчёта мест.
* Администраторы системы должны иметь возможность добавлять, изменять и удалять мероприятия из списка по мере необходимости, а также просматривать количество регистраций и информацию об участниках.

## 2.2 Выбор фреймворка

Что касается непосредственно стека разработки, был выбран язык программирования Python, поскольку он более разнообразен и универсален, чем низкоуровневые Java и C++ или интерфейсный JavaScript. Такой инструмент не потребует налаживания связей между разношёрстными компонентами и поможет бесшовно и в едином стиле реализовать как обращения к базе данных, так и отрисовку интерфейсов. Более того, как популярный на данный момент и вместе с тем всё ещё активно развивающийся язык, он обладает огромной базой документации и поддержки, как официальной, так и предоставленной сообществом разработчиков.

Целью сделать систему портативной и легко встраиваемой была продиктована необходимость реализовать продукт в виде веб-приложения. Это породило вопрос о выборе окружения (фреймворка, framework) для веб-разработки, которых Python предоставляет несколько в силу своей многогранности. Основной выбор заключался между Flask и Django, исследование информации о них показало, что второй может быть назван более предпочтительным для моего формата работы благодаря своей цельной, «жёсткой» структуре объединения компонентов [1].

## 2.3 Выбор СУБД

Для реализации поставленной задачи было также необходимо организовать методику обработки и хранения данных, что поставило меня перед выбором системы управления базами данных. Планируемые структуры данных имеют чёткий формат и прямую взаимозависимость, поэтому в терминах базовой классификации выбор сразу пал на реляционную СУБД, использующую таблицы и поддерживающую язык SQL. При выборе между конкретными вариантами предпочтение было отдано PostgreSQL за счёт таких его преимуществ, как быстрое масштабирование, наличие множества расширений стандартной функциональности по сравнению с другими подобными базами данных и, наконец, эффективная интеграция с выбранным ранее Django [2].

# 3. Описание программного продукта

## 3.1 Общая файловая структура

Ссылка на открытое хранилище с исходным кодом прототипа продукта приложена в общей пояснительной записке.

Основа файловой структуры обоснована строением классического проекта во фреймворке Django, который, как уже упоминалось, предпочитает навязывать разработчикам свои рамки. Впрочем, за счёт этого контролируется целостность взаимодействия компонентов, поэтому меня это вполне устроило. Рассмотрим структуру пакета: скрипт *manage.py* служит для глобальных операций над продуктом, таких как запуск из командной строки или применение значительных структурных изменений. Директория *reg\_sys\_cw* содержит, опять же, файлы, контролирующие глобальные стороны проекта: из тех, с чем я непосредственно взаимодействовала, *settings.py*, осуществляющий контроль над параметрами настроек, и *urls.py*, связывающий компоненты кода с адресами в браузере.

Работа в основном была сосредоточена в директории *regsys*. Разделы *templates* и *static*, относящиеся к визуальному оформлению интерфейса, будут подробнее рассмотрены в индивидуальной работе коллеги, я же в следующих параграфах сконцентрируюсь на содержащихся здесь Python-файлах.

## 3.2 Структура базы данных

Наглядно изображённая схематически структура сущностей и отношений в используемой базе данных: [3]

Используется три основных сущности: **Событие** (**Event**), **Расписание** (**Timetable**) и **Участник** (**Guest**). **Событие** используется для внешних событий, которые включают в себя более мелкие мероприятия и могут длиться несколько дней – например, Зимняя школа факультета компьютерных наук или День открытых дверей. **Расписание** – термин может казаться не совсем логичным, но был выбран во избежание путаницы – включает в себя уже мелкие мероприятия, такие, как конкретные лекции и мастер-классы. Между первой и второй упомянутыми сущностями установлена связь «один ко многим», которую отражает внешний ключ в поле **Timetable.event**.

**Участник** отражает зарегистрировавшихся пользователей. Наконец, между **Расписанием** и **Участником** реализована связь «многие ко многим», которая представлена в виде вспомогательной таблицы **Запись** (**Registration**). Элементы последней представляют собой пары значений из этих двух таблиц.

Реализованы в СУБД эти таблицы посредством такого инструмента, как Django Модели [4], в файле *models.py*. Для каждой из таблиц он содержит перечисления полей с необходимыми типами и параметрами, а также включает данные им человеко-читаемые имена. Особенно стоит здесь отметить поля, которые нуждаются в дополнительных проверках:

* реализована валидация, сопоставляющая **Event.start\_date** и **Event.end\_date** во избежание неверно указанных дат события;
* подобной же валидацией проверяется, что **Timetable.date** входит в рамки своего «родительского» события;
* с использованием механизма Django Сигналов [5] устроено автоматическое пересчитывание свободных мест при добавлении или удалении **Записи**.

Все вносимые в базу данных изменения контролируются фреймворком в директории *migrations* [6].

## 3.3 Запросы и передача данных

Перемещением между веб-страницами и передачей данных в запросах управляют представления (views), размещённые в файле *views.py*. Каждое из них представляет собой функцию, которая вызывается при переходе по ссылке в адресной строке, выполняет некие действия, возможно, с использованием введённых на прошлой странице данных, и возвращает какой-то браузерный ответ. В данном случае вызываются эти представления ссылками, детализированными в *urls.py*, а после отработки отрисовывают соответствующий им HTML-шаблон (template) из хранящихся в *templates*, передавая все необходимые данные через контекст (context).

Согласно функциональным требованиям, полноценных представлений три, и по названиям они совпадают со своими ссылками. **register** составляет список всех доступных **Событий**, для каждого из них при этом генерируя список дней, в течение которых оно длится, и в итоге отрисовывает это в виде первого, приветственного экрана. Соответственно, **timetable**, разбирая информацию из пришедшего к нему по нажатию кнопки в интерфейсе POST-запроса, создаёт и регистрирует в базе данных нового **Участника**, а затем составляет расписание тех дней, которые только что указали необходимыми. **completed** практически по такому же принципу составляет список мероприятий для показа в финальном списке, но в первой половине работы он создаёт новые **Записи** согласно пришедшему списку регистрации.

Реализовано также четвёртое представление, **download**, которое отличается от остальных форматом возвращаемого объекта. Для него не предусмотрено визуального шаблона – он возвращает сгенерированный файл. Эта функция отвечает за составление таблицы расписания, которая доступна пользователю для скачивания после окончания процедуры.

## 3.4 Панель администрирования

Наконец, необходимо рассмотреть и вторую сторону продукта, чуть более скрытую, но столь же, если не более, важную – администрирование системы. По сути, её разработка не имеет смысла, если не предусмотреть организацию эффективного управления материалом. К счастью, Django по умолчанию предоставляет такой инструмент, как панель администрирования сайта – в отличие от клиентского наполнения, которое может быть невероятно многогранным и которое я обсуждала до этого, сторона менеджмента в подобных проектах довольно похожа, что и позволяет фреймворку сразу же предлагать эффективный и надёжный метод работы [7].

Раздел администрирования реализован «из коробки» как закрытый сервис, вход в который требует логина и пароля - в то время как добавить нового пользователя-администратора требует одного действия от уже зарегистрированного менеджера, сервис защищён от «прохожих» посетителей системы регистрации. Функционал этого раздела представляет собой, по сути, визуальную обёртку над имеющейся базой данных. Для каждой из имеющихся таблиц администратору доступен список объектов, включая возможность их редактировать, добавлять и удалять. И реализовано это не так, как бывает непосредственно на серверах СУБД – что понятно лишь профессионалу – а вполне наглядно и понятно, со множеством удобных кнопок, специфичных окошек и мягким отловом ошибок, словом, подходяще даже для человека, не имеющего программистского опыта.

Единственная работа, которую было необходимо проделать над администраторской панелью, - чисто декоративная. Её описание содержит файл *admin.py*: для каждой таблицы там перечислен список полей для отображения и настроены функции, позволяющие быстро фильтровать связанные таблицы, по функционалу сходные с SQL Join [8].

# 4. Заключение

Вывод, который я могу сделать в своей индивидуальной работе, совпадает с итогами анализа в общем тексте: я считаю, что данная реализация регистрационной системы вполне удовлетворяет всем пунктам поставленного технического задания и, более того, в целом является продуктом, пригодным как для использования по прямому назначению в виде прототипа, так и для масштабирования и интеграции с реальными системами. Могу также отметить, что, на мой взгляд, выбор инструментов и средств разработки в сфере бэкенда оказался действительно оптимальным и полезным, доказав на деле некоторые критерии, упоминавшиеся в теоретическом анализе этих решений.

# Источники

[1] Flask vs Django: Which Python Web Framework to Use in 2023? 2023. <https://hackr.io/blog/flask-vs-django>

[2] Which Modern Database Is Right For Your Use Case? 2023. <https://www.integrate.io/blog/which-database/>

[3] Registration System: ER Diagram. 2023. <https://www.figma.com/file/cpJIurC4qGIu2jrA6UtA9I/Registration-System%3A-ER-Diagram?type=whiteboard&node-id=0-1&t=jskhNTiJYU6o0RCU-0>

[4] Models: Django Documentation. 2023. <https://docs.djangoproject.com/en/4.1/topics/db/models/>

[5] Signals: Django Documentation. 2023. <https://docs.djangoproject.com/en/4.1/topics/signals/>

[6] Migrations: Django Documentation. 2023. <https://docs.djangoproject.com/en/4.1/topics/migrations/>

[7] The Django admin site: Django Documentation. 2023. <https://docs.djangoproject.com/en/4.1/ref/contrib/admin/>

[8] Customize the Django Admin With Python. 2023. <https://realpython.com/customize-django-admin-python/#providing-links-to-other-object-pages>