

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

**Факультет информатики, математики и компьютерных наук
Образовательная программа «Программная инженерия»**

СОГЛАСОВАНО

Руководитель, к. ф.-м. н, доцент
кафедры информационных систем и
технологий факультета информатики,
математики и компьютерных наук

_____ Н.В.Асеева
«___» _____ 2024 г.

УТВЕРЖДАЮ

Академический руководитель
образовательной программы
«Программная инженерия»

_____ Н.В.Асеева
«___» _____ 2024 г.

Выпускная квалификационная работа – БАКАЛАВРСКАЯ РАБОТА

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

по направлению подготовки 09.03.04 «Программная инженерия»

Исполнитель:
студент группы 20ПИ1

_____ Ю.А. Лапшина
«___» _____ 2024 г.

Нижний Новгород, 2024

Аннотация. Данная выпускная квалификационная работа рассматривает процесс разработки и развёртывания электронной системы, предназначенной для регистрации на мероприятия нижегородского кампуса НИУ ВШЭ. Актуальность работы заключается в общей цифровизации общества, нуждающегося в эффективной автоматизации монотонных процессов, а также в решении существующей на факультете практической задачи. Проект реализовывался двумя студентами, в задачи этой конкретной работы входит анализ рынка конкурентов, а также обзор технической стороны проекта – базы данных, алгоритмов работы с данными, архитектуры веб-сервиса, безопасности и разрешений, обслуживания сервера. Проект полностью реализован согласно разработанной спецификации, существует в открытом доступе и готов к использованию. Ссылки на хранилище исходного кода и документации, а также на сам веб-сервис доступны в Приложениях к данной работе.

Ключевые слова: система регистрации, организация мероприятий, веб-приложение, Python, Django

Содержание

Содержание	3
Введение	5
1. Подготовка к разработке	7
1.1. Обзор существующих аналогов	7
2. Средства и технологии разработки	9
2.1. Язык и фреймворк	9
2.2. СУБД	9
2.3. Сервер и хостинг	9
3. Описание программного продукта	11
3.1. Общая структура файлов	11
3.2. Архитектура базы данных	12
3.3. Организация веб-страницы	13
3.4. Личный кабинет	14
3.5. Основной функционал регистрации	16
3.6. Генерация файлов	17
3.7. Электронная почта	18
3.8. Панель администрирования	20
3.9. Настройки и безопасность	21
4. Эксплуатация программного продукта	24
4.1. Размещение на сервере	24
4.2. Техническая документация	24
4.3. Версии и перспективы	25
Заключение	26
Источники	27
Приложения	29
Приложение 1. Сравнительный анализ аналогов продукта	29
Приложение 2. Хранилище исходного кода	30
Приложение 3. Архитектура базы данных	31
Приложение 4. Страница регистрации	32
Приложение 5. Страница авторизации	33
Приложение 6. Страница профиля	34
Приложение 7. Страница мероприятий	35
Приложение 8. Страница уточнения расписания	36

Приложение 9. Список мероприятий участника	37
Приложение 10. Генерация расписаний	38
Приложение 11. Генерация сертификатов	39
Приложение 12. log_info.log	40
Приложение 13. Схема обмена электронной почтой	41
Приложение 14. Почтовый диалог с пользователем	42
Приложение 15. Администрирование – страница просмотра	43
Приложение 16. Администрирование – страница изменения	44
Приложение 17. Система регистрации в общем доступе	45
Приложение 18. apache/sites-available/django.conf	46

Введение

Информатизация – проникновение технологий в не связанные с ними напрямую сферы общества – в современном обществе достигла настолько широкой степени распространения, что во многом стала уже привычной. Всё чаще люди ожидают от окружающих их товаров и услуг соответствия критериям цифровой сферы: широкой доступности информации и высокой скорости процессов. Эти характеристики влекут за собой эффективность и комфорт, а следовательно, повышают стандарт жизни – освобождают для более необходимых дел такие ценные ресурсы, как силы и время. Причём важно их обеспечение не только для потребителей, для которых программный продукт – лишь мост между одной конкретной потребностью и её удовлетворением. По сути, для производителей, которые весь свой профессиональный процесс организуют через этот же продукт, его удобство играет чуть ли не большую роль. Ведь автоматизация процессов – это существенная экономия времени сотрудников и бюджета компании, не говоря уж о таких аспектах, как уменьшение вездесущего фактора человеческой невнимательности [1].

Линейка процессов, которые могут подвергнуться эффективной автоматизации – достаточно объёмных по клиентскому массиву, но при этом простых и однообразных по сути – очень широка, но не последнее место в ней занимает практическая задача регистрации на мероприятия. Такой несложный алгоритм, как сопоставление человека с событием, при реализации вручную может наткнуться на множество препятствий: собирающий заявки менеджер их растеряет или неправильно посчитает, а желавшие прийти люди не потрудятся запомнить место и время встречи, которое потом будет непонятно, где выяснить. Словом, наличие систематизирующего – и, что немаловажно, охраняющего – всю эту информацию инструмента значительно облегчило бы жизнь всем участникам процесса.

Необходимость в подобном решении возникла и у нижегородского кампуса Высшей школы экономики. Университет регулярно проводит всевозможные мероприятия, от небольших встреч студенческих клубов до огромных праздников и серьёзных соревнований, гостями которых выступают потенциальные абитуриенты. Но единой системы регистрации при этом не существует, и по необходимости каждое подразделение ищет собственные, сторонние и не всегда понятные инструменты сбора информации, что затрудняет интеграцию данных для получения общей аналитической картины. Кроме того, отсутствуют способы автоматизированного контроля гостей на месте, что создаёт существенные проблемы с подсчётом размера аудитории и подтверждением посещения. Именно в необходимости рационализировать упомянутые процессы, чтобы упростить организационную работу факультетов и улучшить впечатления посетителей, и заключается актуальность нашей работы.

Исходя из вышесказанного, целью данной выпускной квалификационной работы стала разработка системы, которая могла бы использоваться как для

регистрации гостей на проводимые мероприятия, так и для управления ими со стороны менеджеров факультета; для обеих сторон взаимодействия основными критериями, которые планировалось достичь, были универсальность и гибкость, а также наглядность и простота в использовании.

Основными этапами, пройденными по мере выполнения работы, стали:

- сбор требований к продукту, описывающих существующие потребности, и их формализация в виде технического задания;
- анализ присутствующих на рынке приложений аналогичного назначения и их сравнение с предполагаемым продуктом;
- планирование и распределение ролей в проекте, проведённое на основе имеющегося профессионального опыта и желания развиваться в дальнейшем;
- выбор инструментов и технологий разработки в соответствии с поставленными целями;
- проектирование и выстраивание архитектуры продукта;
- разработка всех аспектов продукта – серверная основа, алгоритмы обработки данных, внешние интерфейсы;
- тестирование прототипа системы, проверка на соответствие заявленным требованиям;
- выпуск продукта на открытую платформу для использования широкой аудиторией;
- фиксация проделанной работы в технической документации, руководствах пользователя и непосредственно в тексте работы.

Работа над описанным проектом велась двумя студентами: автором этой выпускной квалификационной работы, а также Титовой Надеждой Дмитриевной, представляющей ту же учебную группу. Сферы ответственности в проекте были чётко разграничены на протяжении всех этапов сотрудничества. Непосредственно в этой работе рассматриваются следующие задачи:

- на подготовительном этапе – обзор существующих на рынке продуктов подобной категории;
- на основном этапе – реализация серверной и алгоритмической части системы (бэкенд, backend), включая описание выбранных для разработки инструментов и подробное описание структуры программных компонентов.

Курсовая работа третьего курса была посвящена той же теме и представляла собой начальные стадии этого же проекта, поэтому некоторые главы текста могут перекликаться с прошлогодней пояснительной запиской [2]. Объём выполненной работы на тот момент включал анализ рынка, а также реализацию лишь самых базовых алгоритмов обработки данных, использовавшихся в системе. За период работы над выпускной квалификационной работой проект был заметно переработан: расширена структура, добавлено множество мелкого функционала, а также, разумеется, осуществлено развёртывание на открытую платформу.

1. Подготовка к разработке

1.1. Обзор существующих аналогов

Было проведено изучение и сравнение имеющихся сейчас на рынке программного обеспечения продуктов, предназначенных для решения схожих задач. Наглядная таблица сравнительных характеристик представлена в Приложении 1.

Основными критериями стали типы мероприятий, при организации которых эффективно применять продукт.

- По количеству ожидаемых гостей события можно разделить на большие и малые – граница, конечно, условна, но для примера поставлена на сотне человек. Для наглядности можно сформулировать следующим образом: малыми можно назвать события, на которых предполагается, что гости более-менее друг друга знают и готовы общаться друг с другом, а не просто слушать одного оратора – например, личные праздники.
- Затем, по частоте события можно подразделить на регулярные – проводящиеся не реже раза в месяц – и нерегулярные, случающиеся реже.
- Существует также возможность разделения по формату, на онлайн-события, живые встречи и гибридные формы; однако в контексте исследования это оказалось заметно менее важным, поскольку не несёт значительной разницы в функционале продукта, так что было опущено из сравнения.
- Наконец, отдельной категорией следует вынести возможность организовывать мероприятия с ограниченным количеством мест, чтобы форма регистрации динамически контролировала доступность и сообщала о ней пользователю.

Среди критериев другого рода были выделены [3]:

- Дополнительные возможности функционала: для пользователя - возможность видеть все регистрации в едином удобном расписании, для организатора – возможность собирать в единую базу контактную информацию пользователей и видеть количество посетителей, а также возможность организации единой почтовой рассылки, будь то реклама или организационная информация.
- Оценка стоимости использования сервиса, как для организатора, так и для посетителя. При исследовании выделились три основных паттерна: сервис по модели ежемесячно оплачиваемой подписки; сервис с пропорциональной комиссией, что означает бесплатное использование при продаже бесплатных билетов и, соответственно, рост цен на обслуживание при подорожании самого события; и сервис без дополнительной комиссии. Нужно понимать, однако, что при любой из этих схем, даже при «условно бесплатной» третьей, организатор мероприятий всё равно будет вынужден нести некоторые издержки на поддержку программного обеспечения – тратить время и заработную плату на разработку, установку и наладку инструментов.
- Платформа, на которой реализован продукт – веб-страница, десктоп-

приложение для компьютера или мобильное приложение – то есть доступность и удобство использования.

Из шести продуктов, исследованных, как самые широко распространённые и известные на рынке, три можно выделить в единую группу. Eventbrite, Whova и Bizzabo - это коммерческие инструменты регистрации, которые характеризуются высокой гибкостью относительно мероприятий – то есть подходят под большинство критериев первого списка – и ориентированностью на собственные приложения пользователя, будь то ПК или смартфон [4]. С другой стороны, у них, естественно, чаще встречается плата за использование, по любой из двух названных схем. Продукт Wild Apricot используется в более узкоспециализированных ситуациях: он наиболее эффективен для регулярных встреч, причём для некоммерческих и социальных организаций не очень большого объёма, применение его для других задач возможно, но более затратно [5].

Оставшиеся сервисы представляют собой доступные для непрофессионалов инструменты: для них характерно отсутствие пользовательского личного кабинета, что позволяет применять их для широкой незнакомой аудитории, и некоторых сложных нюансов функционала, например, динамического подсчёта доступности. Эти сервисы чаще встречаются в чистой форме веб-страниц, что, опять же, максимально упрощает и ускоряет работу с ними, а также делает проще их интеграцию в сторонних сайтах и социальных сетях. Они также обходятся заметно дешевле, если не совсем бесплатно. Стоит, однако, заметить, что при всей описанной простоте Google Forms доступны для расширения функционала: при наличии персонала с навыками программирования организаторы могут сделать инструмент более гибким и подходящим под их потребности посредством работы с Google Forms API [6]. Timepad, к сожалению, такой возможности не предоставляет.

Что касается планируемого для разработки продукта, основными его задачами является скорость настройки, удобство как для организаторов, так и для желающих зарегистрироваться пользователей, простота интегрирования в имеющиеся ресурсы университета и, разумеется, выгода для обеих сторон. Именно поэтому форма реализации пока будет уступать приложениям первой группы, но по функционалу поставит себе цель приблизиться к ним – и, на данный момент, сохранит бесплатное использование, пусть и не отрицая возможности когда-нибудь расшириться до полноценного коммерческого продукта.

2. Средства и технологии разработки

2.1. Язык и фреймворк

Первым выбором касательно набора технологий – стека – разработки предстояло быть языку программирования и, возможно, какому-то его более специализированному окружению (фреймворку, framework). Основной формой реализации для продукта было выбрано веб-приложение – это объясняется требованиями к его гибкости и доступности. Такое приложение, в отличие от мобильного или от десктоп-формы, можно намного быстрее сделать доступным на широком ряде устройств – не требуется никаких скачиваний и установок, достаточно иметь браузер и Интернет-соединение. Достичь такой цели можно было несколькими путями – например, такие низкоуровневые языки, как C++ или Java, хорошо подошли бы для бэкенда, тогда как JavaScript обеспечил бы отличный интерфейс. Выбран, однако, был Python – потому, что является крайне гибким и многогранным, благодаря чему все названные компоненты могут быть реализованы бесшовно, без необходимости решать задачи склейки разношерстных компонентов.

Что касается выбора непосредственно фреймворка для веб-разработки, двумя самыми распространёнными инструментами в этой сфере являются Flask и Django. Исследование информации о них показало, что второй может быть назван более предпочтительным для планируемого формата работы благодаря своей цельной, «монолитной» структуре объединения компонентов, обеспечивающей надёжность и безопасность [7].

2.2. СУБД

В основе работы системы лежат алгоритмы сбора, хранения и обработки данных, поэтому сложно переоценить важность выбора подходящего способа их управлением. Из требований универсальности и эффективности работы системы вытекает вывод, что наиболее удобно будет использовать систему управления базой данных со строгой внутренней структурой и управляемую языком SQL, иначе говоря, реляционную. При выборе конкретного варианта самым подходящим был сочтён PostgreSQL – среди решающих преимуществ можно выделить расширенный инструментарий, масштабируемость и удобство интеграции с выбранным ранее Django [8].

2.3. Сервер и хостинг

Процесс выпуска продукта на открытую реальному миру платформу (релиза, release) также потребовал принятия множества решений относительно используемых инструментов. Прежде всего, предстояло решить, какая будет использоваться стратегия развёртывания на сервере (деплоя, deploy). В числе наиболее часто предлагаемых можно назвать «платформу как сервис» (PaaS), при которой вся сложная и требующая специальных знаний работа над серверной

частью делегируется специальным сервисам, а также использование виртуального приватного сервера (VPS), при котором вся необходимая работа становится ответственностью разработчика, а всеми ресурсами он управляет самостоятельно в облачном аккаунте. В результате была выбрана стратегия деплоя вручную, которая похожа на вторую названную с той только разницей, что управление происходит на реальной, собственной машине. Преимущества этого подхода перед PaaS – гибкость настройки и больший контроль, углубление знаний и умений в данной сфере работы, а также финансовая экономия [9].

Непосредственно выбор веб-сервера для деплоя вручную заключался между такими популярными инструментами, как Apache и NGINX. Предпочтение было отдано первому за счёт его высокой гибкости и широкого набора модулей [10]. Наконец, необходимо было также самостоятельно обеспечить продукту доступное доменное имя. Основным критерием в данном вопросе был, опять же, финансовый – на некоммерческое, тестовое размещение проекта не очень хочется выкидывать много денег. Качественное доменное имя можно было бы получить при использовании PaaS-услуг, но, так как от них было решено отказаться, для решения данного вопроса был использован хостинг, бесплатно предоставляющий ограниченные тестовые возможности мелким проектам – CloudDNS [11].

3. Описание программного продукта

3.1. Общая структура файлов

Исходный код регистрационной системы располагается в хранилище GitHub, вместе с ним доступна пошаговая история планирования и выполнения работы в виде GitHub Issues, а также техническая документация в виде GitHub Wiki. Ссылки в виде текста и QR-кода можно найти в Приложении 2.

Пояснение текстовых стилей, использованных в этой и следующей главах: **полужирным** выделены названия сущностей базы данных, *курсивом* – названия файлов и папок, а также названия функций, классов и других элементов кода.

Как уже упоминалось, Django придерживается монолитного подхода к разработке и предпочитает навязывать пользователям свои готовые решения. Тем не менее, это сложно назвать недостатком – наличие готового шаблона для какого-то функционала ничуть не ограничивает простор для его кастомизации, то есть расширения на свой вкус. Особенно же кстати это приходится на начальных этапах разработки, где вместо того, чтобы по кусочкам собирать основу программы, ты получаешь уже работающий минимум. Поэтому файловая структура проекта была чётко определена фреймворком с самого начала. Коротко рассмотрим пункты, которые можно увидеть в хранилище исходного кода:

- *.idea/* – системная папка, содержащая информацию о структуре входящих в проект модулей.
- *documents/* – не относится непосредственно к коду. В этой папке хранятся все отчёты и пояснительные записки, которые на протяжении двух лет сдавались для защиты проектов на данную тему. При желании можно убедиться, что прогресс действительно имеет место, и значительное.
- *reg_sys_cw/* – содержит системные файлы, контролирующие взаимодействие проектных модулей. Файл *settings.py* будет упомянут в более подробном описании алгоритмов.
- *regsys/* – папка, содержимое которой описывает уже непосредственно модуль регистрационной системы. По сути, именно её разбору посвящена вся глава 3 (поэтому префикс папки может быть опущен в дальнейших разделах).
- *static/* и *templates/* (а также *regsys/static/* и *regsys/templates/*, которые, по сути, друг друга дублируют) – места хранения статических файлов и шаблонов интерфейса соответственно. Об их использовании также будет рассказано здесь, но за более подробным описанием их структуры и разработки стоит обратиться к работе коллеги, посвящённой фронтенду, то есть работе над интерфейсной стороной системы.
- *.gitignore* – файл, управляющий сохранением конкретных вещей в системе контроля версий. Обеспечивает сохранность данных, которые необходимо держать вне общего доступа - подробнее в разделе о безопасности, – а также просто порядок, позволяя не сохранять неважные для работы файлы.
- *README.md* – согласно правилам хорошего тона, записка для первого

ознакомления с проектом.

- *manage.py* – системный скрипт, контролирующий глобальную работу проекта и позволяющий проводить над ним операции в терминале, например, взаимодействовать с внутренним сервером или с напрямую базой данных. Здесь не освещается подробно, поскольку реализован самим фреймворком, но функционал описан в технической документации.
- *requirements.txt* – список модулей, которые требуется установить для нормальной работы проекта.

Кроме того, в структуре проекта есть, как уже упомянуто, файлы, не отслеживаемые на GitHub по причинам либо безопасности, либо просто отсутствия необходимости в версионировании. Тем не менее, они присутствуют непосредственно в рабочем пространстве, так как важны для программы. Они включают папку *django_env/*, содержащую виртуальное Python-окружение проекта, а также два файла, которые тоже будут позже рассмотрены подробнее: лог системы и *.env*.

3.2. Архитектура базы данных

Наглядно изображённая структура сущностей и отношений в используемой базе данных размещена в Приложении 3.

Основным хранилищем информации о проводимых мероприятиях являются модели **Событие** и **Расписание**. **Событие** может быть представлено такими примерами, как «Зимняя школа-2024» или «День магистратуры», то есть глобальными мероприятиями, охватывающими несколько более мелких активностей. Для таких есть смысл хранить, не считая названий и аннотаций, только общие границы времени и места – первый и последний день, а также глобальный адрес, например, конкретный корпус университета. **Расписание** же (название может показаться не самым интуитивным, но выбрано ради избежания путаницы в синонимах) описывает именно мелкие активности в этих списках – конкретные лекции и мастер-классы. Для них уже важно включить день и временной слот, номер аудитории, ведущего, количество мест и, собственно, принадлежность к конкретному **Событию**. Можно упомянуть два поля, работа с которыми происходит особым образом. Первое – это счётчик занятых мест, который, в отличие от общего их количества, недоступен к редактированию и самостоятельно изменяется в зависимости от происходящих регистраций. Второе – это флажок «Повтор?», задачей которого является отслеживать повторение активностей в расписании. Он автоматически помечает повторяющимися элементы **Расписания**, имеющие одинаковое название и принадлежащие к одному и тому же **Событию**.

Информация о пользователях системы управляется такими сущностями, как **Пользователь** и **Участник**. Первая представляет собой системную модель, автоматически создаваемую и контролируемую системой логинов Django [12]. Она содержит логин и пароль, который хранится с максимальной безопасностью, а также информацию, касающуюся разрешений пользователя – на просмотр

определённых страниц, редактирование записей базы данных и так далее. Эта модель, в свою очередь, расширяется моделью **Участник**, где хранится интересующая организаторов личная и контактная информация о каждом посетителе – ФИО, место обучения, номер телефона и логин в Telegram. По желанию пользователь может эти поля не заполнять, на существование и общую работу его профиля это никак не повлияет. Для персонала, обслуживающего систему, дополнительная запись **Участника** не создаётся – достаточно служебной по умолчанию.

Результатом непосредственно алгоритма регистрации становится связь типа многие-ко-многим между **Участником** и элементом **Расписания** – в данной архитектуре она представлена как сущность **Запись**. Кроме двух названных ключей, она также содержит информацию о статусе регистрации: для предстоящих мероприятий может быть «Ещё не посещено» или «Пересекается», а для прошедших – «Пропущено» и «Посещено».

Также на уровне базы данных реализована система лейблов – она призвана упростить поиск и изучение мероприятий, сделав их фильтруемыми по определённым критериям. **Тег** предоставляет место для хранения всех используемых для этого лейблов, каждый из которых должен содержать категорию, например, «Целевая аудитория», и непосредственно название, например, «Для школьников». Привязка **Тега** к **Событию** осуществляется через **Теги событий**.

На уровне кода все эти сущности реализованы не напрямую в базе данных, а через функционал Django-моделей [13]. Они описаны в файле *models.py* как дочерние классы *Model*, с перечислением всех полей и небольшими вспомогательными функциями вроде проверки допустимых границ и возвращения человеко-читаемого названия. Автоматические изменения полей элемента **Расписания**, упомянутые выше, также контролируются здесь – функциями с декораторами *receiver*, которые вызываются при сохранении или удалении записей необходимого типа [14]. Фреймворк сам переводит это в команды более низкого уровня и применяет к базе данных, когда происходит применение изменений – так называемая миграция [15]. Генерируемый при этом код на языке SQL сохраняется в папке *migrations/*.

3.3. Организация веб-страницы

Для того чтобы как следует понимать описываемые дальше процессы, необходимо сначала коротко обсудить базовые принципы построения контента. Приложение представляет собой набор веб-страниц, каждая из которых включает следующие слои:

- адрес. Каждая страница имеет свой уникальный определитель, подставляемый после основного доменного имени в адресную строку. Они определяются в файле *urls.py*: список *urlpatterns* содержит все известные программе адреса, сопоставленные с соответствующими им функциями представления. Любой

адрес, соответствие которому в этом файле не нашлось, вызовет появление ошибки «404: Страница не найдена».

- представление, или *view*. Поставленные в соответствие адресам функции имеют вид *views._name_* и определены, следовательно, в файле *views.py*. Эти функции принимают на вход запрос на открытие определённой страницы и, собственно, открывают её, между этими двумя пунктами проделывая все необходимые операции с данными [16]. Это относится и к данным, пришедшим на вход как часть отправленного GET- или POST-запроса, и к данным, необходимым для передачи новой странице. Последние собираются в специальный словарь *context* и на выходе передаются в функцию *render*.
- шаблон, или *template*. Некоторые служебные функции-представления заканчиваются перенаправлением на другие страницы, но большинство служит для открытия своей собственной, и возвращают они результат функции *render* с упоминанием собранного контекста, а также некоторого файла с расширением *.html*. Это и есть шаблон – описание внешнего вида финальной страницы, в который на финальной загрузке подставляются все необходимые переменные. Эти файлы хранятся в папке *templates/* и написаны на специальном языке шаблонов Django [17], представляющем собой HTML-разметку, расширенную функционалом для добавления Python-подобных команд вроде переменных, условий, циклов и операторов. Большая часть используемых команд уже реализована самим фреймворком, но для определения собственных операторов также используется файл *templatetags/extra-tags.py*.

В дополнение ко второму пункту списка можно упомянуть остальные элементы файла *views.py*: служебные функции *get_paragraph_style* и *letter_first_cmp*, роль которых упоминается позже, а также функцию *dispatcher*, которая разрешает запросы, не требующие отрисовки собственной страницы – например, удаление или редактирование данных, визуально требующее лишь перезагрузки уже открытого адреса.

Что касается внешней структуры страниц, она в основном проста и выдержана в едином стиле – предоставляемая информация структурируется списками, а для сбора новой построены минималистичные формы с полями для ввода и кнопками отправки. Для эффективной навигации реализована верхняя навигационная панель – на ней закреплены ссылки на все основные разделы системы, сгруппированные по функционалу и важности (*views.navbar_sign* и *views.navbar_profile*). А для своевременного информирования пользователя о результатах системных действий – успехах, предупреждениях и ошибках - реализован функционал вывода на экран сообщений [18]. Визуальные примеры конкретных страниц будут приведены позже.

3.4. Личный кабинет

Знакомство пользователя-гостя с приложением начинается с личного кабинета – ради обеспечения безопасности и цельности процессов архитектура системы выстроена так, что весь главный функционал требует от пользователя выполнить

вход в профиль. Все имеющиеся в приложении страницы можно разделить на доступные и не доступные для не вошедших в профиль «анонимов» - определяется это наличием декоратора *login_required* перед соответствующей функцией-представлением [19]. Верхние навигационные панели для этих двух блоков тоже различны. В «анонимный» блок входят только страницы регистрации, авторизации, восстановления пароля, не считая обратной связи, пользовательской справки и приветствия.

При входе на основную страницу системы, без конкретизации адреса, пользователя встречает приветствие, или лендинг (*views.landing*) – оно кратко поясняет, что к чему, и кнопкой приглашает на регистрацию. Страница регистрации (*views.signup*) представляет собой единственное поле для ввода логина и несколько галочек-чекбоксов, чтобы пользователь подтвердил своё согласие на обработку персональных данных и фотографирование на мероприятиях. От гостя не требуется сразу предоставлять все личные и контактные данные, но в то же время хочется иметь уверенность, что в случае чего у менеджеров будет хоть один подтверждённый канал связи с ним – из этих соображений логином в системе является электронная почта. Помимо перечисленных плюсов, у этого есть ещё одно важное преимущество с точки зрения пользовательского опыта – специально выдуманные для сайта логин и пароль легко забыть, чего не скажешь об используемом адресе электронной почты и высланном на него пароле.

Итак, после заполнения этой формы (*views.dispatcher, sender="signup"*) создаётся новый профиль пользователя, и на введённую пользователем электронную почту высылается сгенерированный системой пароль (подробнее о работе рассылок – в соответствующем разделе). Вход при этом не осуществляется автоматически – то, что пользователь должен вручную найти и ввести свой пароль, служит самым эффективным подтверждением почты из возможных. На странице авторизации (*views.signin*), очевидно, можно осуществить вход по имеющимся логину и паролю. Страница восстановления пароля (*views.forgot*) тоже предлагает ввести адрес почты – в случае, когда такой аккаунт в системе действительно есть, на почту высылается новый пароль. В случае всевозможных ошибок на этих страницах – ввода неверного пароля, отсутствия аккаунта и так далее – пользователя информируют о случившейся ошибке интерфейсные сообщения. Страница справки (*views.help_anon*) рассказывает о том, как всем этим пользоваться (подробнее о дизайне – в работе коллеги), а страница обратной связи (*views.feedback_anon*) позволяет связаться с менеджерами (подробнее – в разделе о почте). Описанные страницы продемонстрированы в Приложениях 4 и 5.

После того, как вход в профиль осуществлён, пользователю становится доступен уже основной блок страниц, рассматривать который мы начнём с управления профилем. Страница профиля (*views.profile*) предоставляет две формы. Одна из них предназначена для просмотра и обновления личных и контактных данных (*views.dispatcher, sender="edit_info"*) - как уже упоминалось, они не обязательны для заполнения, поэтому менять их здесь можно сколько угодно раз и по сколько угодно полей. Организатор может при желании попросить гостей определённых мероприятий заполнить ту или иную информацию, а также ФИО

необходимо для генерации личных подтверждающих документов – в остальном здесь достаточно свободная политика. Вторая же форма предназначена для управления данными авторизации (*views.dispatcher, sender="edit_creds"*). Смена почты из-за архитектуры системы не предусмотрена, но смена пароля возможна и проходит по достаточно стандартному, в целом, алгоритму. Страницу профиля можно увидеть в Приложении 6.

Выход из профиля (*views.signout*) можно осуществить по нажатию выделенной красным кнопки «Выйти» в панели навигации.

3.5. Основной функционал регистрации

Алгоритм, лежащий в основе решения задачи продукта, берёт своё начало на странице «Мероприятия» (*views.register*). На ней пользователя встречает список **Событий**, каждому из которых соответствует своя кнопка регистрации. Кроме этого, в левой части страницы присутствует панель фильтров, где можно выбрать только необходимые **Теги**; после фильтрации страница перезагрузится, оставив в списке только подходящие мероприятия.

Выбор **События** ведёт на страницу уточнения расписания (*views.timetable*), формирующуюся для каждого из них отдельно и поэтому недоступную с верхней панели ссылок. Здесь пользователю открывается многоуровневый список: все принадлежащие данному **Событию** элементы **Расписания**, сгруппированные сначала по дням, а затем по категориям – то есть временным слотам, или частям дня, когда они проводятся. Возможны ситуации, когда одновременно проводятся несколько активностей, но записаться, естественно, можно только на одну – поэтому каждая категория представляет собой набор значений для радио-кнопки. Пользователь может тщательно изучить весь список, чтобы правильно расставить приоритеты с учётом всех пересечений, и наконец, завершить регистрацию, выбрав по варианту в каждой из предоставленных категорий.

Функция *views.completed* обработает запрос на регистрацию, после чего гость вернётся на страницу «Моё расписание» (*views.mylist*), являющуюся для системы страницей по умолчанию. На ней доступны для просмотра все элементы **Расписания**, на которые у данного **Участника** существуют **Записи**. Они разделены на два блока: предстоящие и прошедшие. Отображение самого списка от этого не зависит, но варьируются дополнительные действия, кнопки которых отрисованы рядом. С прошедшими **Событиями** можно сделать только одно – сформировать PDF-сертификат, подтверждающий посещение. Для предстоящих можно скачать расписание в формате удобного списка, а для каждой конкретной Записи в них формируется QR-код. Здесь можно также изменить регистрацию – в этом случае система снова направит пользователя на страницу уточнения, заранее проставив кнопки уже выбранных элементов **Расписания** – или просто удалить (*views.dispatcher, sender="delete"*). Более детально процесс формирования всех перечисленных файлов будет рассмотрен в следующем разделе, а демонстрация страниц находится в Приложениях 7-9.

3.6. Генерация файлов

Как бы удобна ни была какая-либо система, ей редко удаётся обойтись без функционала загрузки информации во внешние источники – ведь часто людям удобнее сверяться со скачанными документами, чем загружать списки на каком-то сайте. Поэтому и в данном приложении ряд процедур предоставляет возможность сгенерировать тот или иной внешний файл. Формирование файлов осуществляется в тех же обрабатывающих запрос функциях-представлениях, а загрузка осуществляется за счёт того, что вместо *render* они возвращают объект *HttpResponse* с указанием типа «приложение» (attachment) и какого-либо конкретного файлового формата.

В трёх алгоритмах системы предусмотрена загрузка документов формата PDF. Один из них – это скачивание справочного файла для администраторов (его содержание описывается в работе коллеги, а про интерфейс администратора будет упомянуто позже). Оно не представляет особенного интереса для анализа, поскольку файл уже свёрстан изначально и не подвергается при скачивании никаким изменениям, происходит оно точно по описанному выше принципу в функции *views.help_admin*. Остальные два случая – это скачивание расписаний и сертификатов о посещении, где уже требуется изменение файлов из кода.

Создание и заполнение PDF-документов с нуля достаточно эффективно решается с помощью Python-библиотеки ReportLab [20]. Функции, в которых используется этот процесс, – *views.certificate* и *views.download* соответственно. Сначала инициализируется объект класса *Canvas* и определяются доступные размеры листа, затем один за другим создаются и отрисовываются блоки текста – объекты класса *Paragraph*. Для создания такого блока используется два аргумента – строка текста, который необходимо вывести, и *ParagraphStyle*, определяющий все его внешние признаки. В целях избежания повторов кода, а также удобного контроля над единством стиля разных документов определение этих стилей вынесено в отдельную функцию – *get_paragraph_style* в том же файле (подробности выбора цветов и шрифтов – в работе коллеги). После определения стиля и содержания блок отрисовывается в документе на передаваемых ему координатах. На примерах конкретно этих документов используется довольно несложное форматирование, расположение блоков просто друг под другом, поэтому для контроля интервалов между ними достаточно изменения одной-единственной переменной, высоты отрисовки.

По завершении работы с одной страницей она переключается методом *showPage*, а весь документ в конце работы сохраняется методом *save*. Генерация расписания на этом завершается, потому что оно целиком состоит из динамически генерируемого списка, но для сертификата нужен ещё один шаг. Этот документ помимо информационной функции несёт в себе до некоторой степени ещё и рекламную – произвести своим дизайном приятное впечатление на получателя. Поэтому дизайн таких сертификатов обычно становится объектом тщательной работы, и именно поэтому система позволяет менеджерам добавлять профессионально созданные PDF-шаблоны сертификатов с тем, чтобы на них

дописывались только имена и названия. Словом, нарисованный холст с динамической информацией теперь нужно «наклеить» на имеющийся фон. Чтение шаблона происходит с помощью другой библиотеки, PyPDF2 [21], а соединение документов - в цикле по страницам с помощью её же метода *merge_page*. На этом работа окончательно заканчивается, и файлы готовы к выгрузке. Полюбоваться примерами таких генераций можно в Приложениях 10 и 11.

Помимо этого формата файлов, приложение использует также генерацию картинок, а именно – QR-кодов. Они, как уже упоминалось, формируются в личном кабинете пользователя для каждой конкретной **Записи**, содержа в себе её ключи. Назначением этих кодов является проверка посещаемости мероприятий – они кодируют системную ссылку, переход по которой меняет статус **Записи** на «посещено». Предполагаемый процесс использования выглядит так: гости открывают свои коды там, где им удобно – с огромной вероятностью, на смартфонах, - а организатор своим гаджетом, большой вероятностью тоже смартфоном, эти коды один за другим считывает, и список пришедших **Участников** автоматически обновляется. Не нужно ни ведения списков, ни опрашивания фамилий – всё может быть решено настолько удобно, и благодаря веб-формату системы достаточно открыть её в браузере смартфона и ввести свои логин и пароль.

Генерация QR-кодов происходит в функции *views.qr_generate* с использованием функционала внешней библиотеки *qrcode*, считывание – вызывает *views.qr_read*. Закодированная ссылка является таким же, как и все прочие ссылки приложения, GET-запросом с ключами **Участника** и элемента **Расписания**, единственное её отличие от большинства уже описанных – декоратор *staff_member_required*, позволяющий выполнить операцию обновления **Записи** только пользователям с менеджерскими правами [22].

Можно упомянуть ещё про одну процедуру, не подпадающую под описанный формат выгрузки, но формально подходящий теме раздела. Фреймворк обладает встроенным функционалом логирования, в рамках которого может сохранять информацию в том числе и в системные текстовые файлы [23]. Настроенный в системе файл *log_info.log* не отслеживается в GitHub ввиду отсутствия необходимости версионирования, но на действующем сервере находится и для демонстрации оставлен в Приложении 12. Чуть больше деталей по настройке логов также будет упомянуто в одном из следующих разделов.

3.7. Электронная почта

Отправление электронных писем является главным способом коммуникации системы и управляющих ей менеджеров с пользователями-участниками. Со стороны кода работы происходит немного: весь функционал, разумеется, уже организован в фреймворке инструментами *django.core.mail*. Объект класса *EmailMessage* инициализируется с указанием таких элементов письма, как тема, отправитель, получатель, тело письма и вложения, а затем, собственно, отправляется.

Общая схема того, как организована работа с адресами отправки и получения, расположена в Приложении 13. Для обеспечения максимального удобства были созданы два аккаунта электронной почты на домене Gmail, которые делят между собой обязанности и служат для немного разных целей. *noreply.hse.regsys* (Бот) является непосредственным представителем приложения – в системе хранится пароль от этого аккаунта, и именно с этого адреса происходит отправка всего автоматически генерируемого контента. Автоматические письма, не требующие ответа – такие, как сгенерированный пароль – приходят напрямую на почту участников. В случае ошибки отправки, не зависящей от работы системы – например, если почтовая служба Google не нашла адрес получателя – сообщение об ошибке приходит обратно Боту. Таким образом, единственными его входящими письмами будут именно ошибки, и поэтому владеть паролем от него достаточно только техническому специалисту, обслуживающему систему.

Второй же аккаунт, *feedback.hse.regsys* (Фидбэк), является «прослойкой», позволяющей вести живое общение с обезличенного официального адреса. Когда пользователь заполняет форму обратной связи, выражая тем самым желание получить ответ, Бот сохраняет обращение в виде входящего письма на Фидбэк. Это письмо содержит адрес участника как значение поля «Ответить», так что ответ непременно найдёт автора. Предполагается, что менеджер, отвечающий за взаимодействие с гостями, будет владеть паролем от Фидбэка и сможет вести структурированные, официальные переписки оттуда без необходимости вводить куда-то адрес собственной рабочей почты. Тем лучше, если таких работников будет несколько, – кто угодно из них сможет продолжить работу другого, когда вся история разговоров хранится в одном месте. Пример переписки, инициированной участником, приводится в Приложении 14.

Чисто системные случаи отправки почты включают в себя уже описанные процессы регистрации и восстановления пароля. Форма обратной связи реализована в приложении в виде двух страниц – в профиле участника и на «анонимной» панели (*views.feedback* и *views.feedback_anon*, соответственно). Единственное их отличие заключается в том, что неизвестному пользователю предлагается ввести адрес электронной почты, на которую он хотел бы получить ответ, тогда как для вошедшего участника таким адресом автоматически считается его логин. В остальном функционал формы достаточно прост – включает в себя поля для темы и для тела письма, а также возможность прикрепить файл формата PDF или изображение.

Что касается менеджерской работы с почтой, то, помимо упомянутых ответов на обращения, для них предусмотрена возможность устраивать массовые рассылки писем для конкретных категорий участников – например, для всех посетителей конкретного мероприятия. Функция *admin.mass_mail* демонстрирует процесс подготовки: формируется список адресов получателей, немного по-разному в зависимости от изначального запроса, а потом по уже описанному принципу письмо собирается и рассылается от имени Бота. Можно заметить, что письмо уходит не напрямую адресатам, а, опять же, Фидбэку с указанием финальных

адресов как «скрытых копий»: это делается затем, чтобы не раскрывать одному получателю весь список остальных.

3.8. Панель администрирования

До сих пор предметом описания в этой главе была та сторона приложения, что предназначена для взаимодействия с пользователем-гостем. Тем не менее, она никогда не сможет использоваться по назначению без второй своей стороны – той, где могут работать управляющие данными менеджеры. Уже был сделан ряд её упоминаний, но описанию уделено чуть меньше места, поскольку очень многое в ней уже было предоставлено самим фреймворком и требовалась лишь небольшая настройка и кастомизация [24]. По сути, панель администрирования (админ-панель) представляет собой автоматически созданный набор веб-страниц для управления моделями, то есть сущностями – как входящим в спроектированную базу данных, так и определённым в системе изначально.

Во второй блок входит упомянутая системная модель **Пользователя**, а также **Группы** – наборы разрешений, которые можно давать определённым пользователям. **Участник** расширяет модель **Пользователя**, следовательно, при регистрации гостей для них вместе с первым создаётся и второй. Но **Пользователь** может существовать и без **Участника** – менеджеры, которым не нужно заходить на основной сайт, вполне могут общаться с админ-панелью именно через такой профиль. Они тоже будут вводить логин и пароль, вдобавок в системе может храниться их ФИО и электронная почта на случай необходимости связи. **Группа** представляет собой некое множество разрешений, объединённое под общим названием, например, «Админ», «Менеджер» или «Помощник». Их создание позволяет поддерживать иерархию среди работников и обеспечивать безопасность – если, допустим, какой-то новичок должен работать только с расписанием и не трогать списки пользователей, чтобы случайно ничего не удалить, это можно обеспечить выдачей ему группы лишь самых базовых разрешений. Полный список разрешений устроен следующим образом: по четыре операции CRUD (create, read, update, delete – создать, прочитать, изменить, удалить) [25] – для каждой модели, включая базу данных и некоторые системные детали.

Ключевое отличие админ-панели от всех остальных страниц приложения в том, что лишь некоторые Пользователи смогут в неё попасть, остальных система попросит авторизоваться под более подходящим логином. Такое право определяется флажком «Статус персонала», присущим **Пользователю** – проставить его положительное значение можно только при создании нового пользователя из админ-панели, но никак не со стороны гостей. Таким образом, новых менеджеров в систему могут добавить только сами менеджеры. Выше упоминалось, что добавить такое свойство странице можно с помощью декоратора `staff_member_required` [22].

Все страницы админ-панели имеют несколько обязательных элементов. На верхней панели представлен набор ссылок управления – можно открыть основной сайт, изменить пароль, выйти из профиля или открыть PDF-справку о работе с

системой. Чуть ниже представлен путь в структуре страниц, так называемые «хлебные крошки» (breadcrumbs) [26]. А слева закреплён список всех моделей, из которого можно перейти как на страницу просмотра, так и на страницу добавления каждой конкретной (типовые примеры можно увидеть в Приложениях 15 и 16).

Эти два типа страниц строятся по одному и тому же принципу для каждой модели, варьируя лишь список соответствующих ей полей. Страница добавления представляет собой просто форму для создания модели – поле ввода значения, название и, при наличии, текстовое пояснение для каждого поля. По заполнению полей (которые, при необходимости, всегда сообщают о проблемах с форматом введённой информации) менеджер сохраняет форму одной из кнопок внизу – с переадресацией обратно в список или с открытием новой пустой формы, в зависимости от его режима работы. Окно для редактирования уже имеющейся записи выглядит и функционирует точно так же, за исключением того, что в полях ввода изначально отображаются старые данные.

Страница просмотра более функциональна. На ней отображается список всех записей данной модели, в каждой из которых ключ выделен и функционирует как кнопка редактирования, а остальные поля представлены в формате таблицы. Отображение этих полей чётко настроено для оптимизации восприятия каждой конкретной модели [27]: например, нет смысла демонстрировать отдельно поля «Всего мест: 20» и «Занято мест: 1», когда для человеческого глаза будет гораздо понятнее прочитать «Мест: 19/20». Там, где это имеет смысл, сформированы поля-ссылки: например, для каждого **События** можно нажать на специальную ссылку, которая откроет список принадлежащих ему элементов **Расписания**.

Для каждой из страниц просмотра реализованы также функции фильтра и поиска (по сути говоря, упомянутые поля-ссылки тоже являются фильтрами, просто сразу сформированными в удобной форме за счёт своей полезности). Поля, участвующие в двух этих вещах, тоже отрегулированы вручную [27]: так, например, фильтровать имеет смысл только признаки, представленные неким небольшим множеством или хотя бы поддающиеся группировке (дата, статус, корпус вуза), а искать – по строкам с высоким разнообразием (фамилии и названия мероприятий). Наконец, существуют действия, которые можно применить к одному или нескольким помеченным галочкой элементам списка – удаление, массовая рассылка почты (*admin.mass_mail*), а также массовое обновление статусов регистрации на «Посещено» на случай, если строгий учёт посещаемости по QR-кодам оказался не критично важен для события (*admin.mark_visited*).

3.9. Настройки и безопасность

Качественная работа системы подразумевает контроль множества глобальных параметров и переменных. Этому посвящён файл настроек – *settings.py*. По большому счёту он представляет собой просто список этих самых переменных, так что его обзор также будет практично построить в виде перечисления. Итак, здесь настраиваются следующие параметры:

- базовая директория *BASE_DIR*. Часто в коде необходимо формировать путь к неким файлам, но вписывать какую-то конкретную ссылку чревато проблемами, поскольку её придётся менять при каждом перемещении программы в файловом дереве, не говоря уж о смене платформы при релизе. Эта переменная позволяет автоматически выяснить расположение текущего файла.
- секретный ключ *SECRET_KEY* используется системой для многих процессов, связанных с безопасностью, включая хэширование, управление сессиями пользователей и генерацию CSRF-токенов [28].
- флажок *DEBUG* сигнализирует, находится система в режиме разработки или реальной работы – от этого зависит несколько основных процессов. Например, в режиме дебага при возникновении ошибки демонстрируется весь стек вызовов, чтобы помочь разобраться с её причинами, а в «боевой» конфигурации пользователя просто встретит аккуратный, но малоинформативный экран ошибки.
- список *ALLOWED_HOSTS* определяет, кто имеет право запускать приложение – здесь прописаны доменный и IP-адрес развёрнутого приложения.
- Блок настроек, относящихся к SSL – контролируют, чтобы вся передача данных шла по защищённому протоколу HTTPS.
- *INSTALLED_APPS* и *MIDDLEWARE* – определяют список модулей, используемых системой. Кроме непосредственно приложения *regsys*, в данном случае достаточно только инструментов самого фреймворка.
- *ROOT_URLCONF*, *TEMPLATES* и *WSGI_APPLICATION* указывают пути к соответствующим элементам приложения – списку адресов, папке шаблонов интерфейсов и настройкам протокола Python-серверного взаимодействия WSGI.
- *CACHES* определяет метод кэширования элементов интерфейса, которое позволяет оптимизировать загрузку страниц.
- блок информации о базе данных – настраивает, под какими параметрами приложение может устанавливать с ней соединение.
- *AUTH_PASSWORD_VALIDATORS* настраивает ограничения на пароль, который можно установить в системе. Кастомный валидатор *RegularValidator* определён в файле *regsys/validators.py*. Навязываемые им ограничения не очень тяжелы, но на шаг выше неконтролируемого пароля – минимальная длина в 8 символов, наличие минимум одной цифры и заглавной буквы.
- настройки электронной почты – перечисляют данные, использующиеся для установки соединений с почтовым сервером.
- настройки локализации – устанавливают часовой пояс и язык региона, чтобы интерфейс (в основном админ-панели) обеспечивал удобное для людей рабочее место.
- *LOGGING* отвечает за формат автоматического логирования системы. Настроены две системы хранения информации – *file_info*, выгружающая в файл *log_info.log* всю информацию уровня INFO и выше, и *mail_admins*, в случае ERROR-сообщения отправляющая развёрнутые отчёты об ошибке на указанные почты технических специалистов [23].

- *ADMINS* – адреса только что упомянутых ответственных за обслуживание приложения.
- *STATIC_ROOT* и *STATIC_URL* - системные адреса, по которым хранятся статические файлы. Метод их загрузки меняется в зависимости от состояния флажка *DEBUG*, а также при некоторых архитектурах систем их может понадобиться хранить вообще на отдельном сервере, так что, опять же, нужен удобный способ динамического контроля этих ссылок.
- *LOGIN_URL* указывает адрес, на который система должна автоматически перенаправлять незнакомых ей пользователей, если для выполняемого ими действия требуется авторизоваться.
- *DEFAULT_AUTO_FIELD* определяет, какой тип по умолчанию имеет поле первичного ключа создаваемой модели.

Ряд из вышеперечисленных настроек (логины и пароли для базы данных и почтового аккаунта, а также секретный ключ) не прописаны напрямую в файле, а запрашиваются функцией *os.getenv*. Эта процедура обеспечивает их сохранность в секрете – благодаря ей данные значения можно хранить в отдельном, зашифрованном от открытого хранилища файле *.env*. Он не приводится тут из формальных требований безопасности, но конструкционно он не представляет собой ничего интересного: так же, как и открытые настройки, включает только несколько переменных и присвоенные им значения [29].

4. Эксплуатация программного продукта

4.1. Размещение на сервере

Для разработки и отладки фреймворк предоставляет множество удобных инструментов во главе с собственным сервером разработки. Достаточно запустить из терминала скрипт *manage.py* с командой *runserver* – и на локальном хосте 127.0.0.1 будет доступна демонстрация сайта. При реальном деплое, однако, такого инструмента уже недостаточно. Как уже упоминалось в главе, посвящённой технологиям, приложение было развёрнуто на сервере Apache2, установленном на локальной машине с операционной системой Ubuntu [30], а база данных организована там же посредством PostgreSQL. Доменное имя зарезервировано благодаря сервису CloudDNS – бесплатный домен третьего уровня hse-reg-sys предоставлен за счёт необходимости использования домена второго уровня dns-dynamic. На данном сервисе были созданы четыре записи типа NS, связывающие приложение с именными серверами, а также запись типа A – привязка IP-адреса сервера – и запись типа CNAME, обеспечивающая корректную переадресацию на сайт при использовании имени с доменом четвёртого уровня www. Текстовую и QR-ссылки на итоговое размещение системы можно найти в Приложении 17.

Помимо описанных внешних процедур, процесс подготовки самого приложения включал лишь смену ряда настроек и конфигураций [31]:

- файлы исходного кода перемещены на нужную машину;
- создано виртуальное Python-окружение проекта и установлены все необходимые библиотеки, перечисленные в *requirements.txt*;
- в файл *.env* внесены актуальные данные для соединения с базой данных;
- список *settings.ALLOWED_HOSTS* обновлён информацией сервера и провайдера;
- обновлён адрес поиска статичных файлов *STATIC_URL*;
- применены команды построения базы данных (*migrate*) и сбора статичных файлов (*collectstatic*);
- сгенерирован SSL-сертификат и добавлены значения настроек, обеспечивающие перенаправление на HTTPS;
- наконец, единственный требующий ощутимой работы шаг в данном списке – для Apache создан файл *sites-available/django.conf*, описывающий взаимодействие сервера и модулей приложения. Его нет в системе контроля версий, но для наглядности он расположен в Приложении 18.

4.2. Техническая документация

Техническая документация – набор текстов, подробно описывающих структуру и процессы системы – доступна по ссылке на хранилище исходного кода (Приложение 2), во вкладке «Wiki». В целом она придерживается той же структуры, что и третья глава данной работы, с добавлением нескольких страниц, касающихся работы с сервером и терминалом. Но содержание построено в более

практическом стиле, с непосредственным цитированием кода и советами по решению всевозможных проблем, которые могут возникнуть в процессе обслуживания системы. Веб-формат документации позволяет легко получать к ней доступ и оперативно редактировать её, а в совокупности с инструментом GitHub Issues и самим контролем версий образуется полноценная и удобная система для работы над продуктом и синхронизации команды.

4.3. Версии и перспективы

На данный момент в хранилище кода можно найти две ветки: master и prod, для удобства их актуальные версии отмечены тегами stable-dev и stable-prod соответственно. Основные их файлы находятся в одинаковом состоянии финального (на данный момент) дизайна системы, разница заключается только в настройках, описанных выше. prod-ветка служит для сохранения текущей работы по обслуживанию, а также как образец для потенциального деплоя на другой платформе; master-ветка может использоваться как подготовительная площадка при последующей разработки нового функционала.

Как было упомянуто в постановке задачи, проект посвящён реально существующей на факультете практической задаче и вёлся под контролем работников вуза, поэтому следующим шагом в работе над проектом является разворачивание приложения как части существующих сервисов ВШЭ. Этот процесс требует длительного согласования и тщательной работы с техническим персоналом вуза, поэтому не выполнен в рамках данной работы; тем не менее, можно сказать, что проведённая работа по деплою на локальном сервере вполне достойно демонстрирует жизнеспособность и цельность продукта. Помимо описанной интеграции в реальные процессы, среди векторов дальнейшего развития проекта можно выделить расширение функционала на основе бета-тестирования и опросов аудитории, а также переход на другие платформы. В текущем состоянии веб-приложения система легко доступна на смартфонах, но должной работы по адаптивированию страниц к мобильной версии проведено пока не было – этот процесс явно стал бы одним из приоритетов развития, а впоследствии вполне мог бы также включить разработку мобильного приложения.

Заключение

Электронные информационные системы, берущие на себя монотонные задачи по сбору и обработке данных, составляют неотъемлемую часть нашего общества, стремящегося к эффективности и удобству. Поучаствовать в разработке подобного инструмента является неоценимым опытом – помимо целого спектра технических навыков, от написания кода до его обслуживания, в данном процессе требуется ещё множество навыков, включая работу с текстовыми спецификациями и документацией, планирование и тайм-менеджмент, общение с заказчиками и с коллегами. Оглядываясь на поставленные задачи, можно сказать, что все планы успешно выполнены – подтверждением тому могут служить результаты тестирования, описанные в работе коллеги. Система регистрации удовлетворяет требованиям, сформулированным руководителями, и в целом является стабильным и самостоятельным программным продуктом, готовым к использованию по назначению. Кроме того, её можно считать вполне конкурентным продуктом, исходя из проделанного анализа рынка схожих решений. Выбор инструментов и технологий вполне оправдал себя – процесс разработки прошёл сравнительно гладко и позволил реализовать всю запланированную архитектуру, процесс деплоя показал успешные и рабочие результаты при минимальной себестоимости. Подводя итог, данную выпускную квалификационную работу можно считать успешно и достойно выполненной.

Источники

[1] Горбунова О.Н. Информатизация общества и формирование трудового ресурса: проблемы, пути решения. Социально-экономические явления и процессы. 2012.

[2] Лапшина Ю.А. Разработка информационной системы для регистрации на мероприятия НИУ ВШЭ – Нижний Новгород. 2023.
https://lms.hse.ru/ap_service.php?getwork&guid=97ECA030-62FD-4A6A-9687-5CD30F6BDD3C

[3] 8 Popular Event Registration Software for Conferences and Events. 2023.
<https://whova.com/blog/event-registration-software-price-comparison/>

[4] Top Rated Event Registration Products. 2023.
<https://www.trustradius.com/event-registration/>

[5] Wild Apricot Membership Management Software. 2023.
<https://www.wildapricot.com/features>

[6] Google Forms API. 2023.
<https://developers.google.com/forms/api/reference/rest?hl=en>

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

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

[9] The Essential Django Deployment Guide. 2023.
<https://www.saaspegasus.com/guides/django-deployment/>

[10] NGINX vs. Apache — Choosing the best web server in 2024. 2023.
<https://www.nexcess.net/blog/nginx-vs-apache/>

[11] Best Free DNS Hosting Providers. 2022. <https://www.keycdn.com/blog/best-free-dns-hosting-providers>

[12] User model: Django Documentation. 2024.
<https://docs.djangoproject.com/en/5.0/ref/contrib/auth/#user-model>

[13] Models: Django Documentation. 2024.
<https://docs.djangoproject.com/en/5.0/topics/db/models/>

[14] Signals: Django Documentation. 2024.
<https://docs.djangoproject.com/en/5.0/topics/signals/>

[15] Migrations: Django Documentation. 2024.

<https://docs.djangoproject.com/en/5.0/topics/migrations/>

[16] Writing Views: Django Documentation. 2024.
<https://docs.djangoproject.com/en/5.0/topics/http/views/>

[17] The Django Template Language: Django Documentation. 2024.
<https://docs.djangoproject.com/en/5.0/ref/templates/language/>

[18] The Messages Framework: Django Documentation. 2024.
<https://docs.djangoproject.com/en/5.0/ref/contrib/messages/>

[19] The login_required Decorator: Django Documentation. 2024.
<https://docs.djangoproject.com/en/5.0/topics/auth/default/#the-login-required-decorator>

[20] ReportLab – PDF Library User Guide. 2024.
<https://www.reportlab.com/docs/reportlab-userguide.pdf>

[21] Welcome to PyPDF2. 2024. <https://pypdf2.readthedocs.io/en/3.x/>

[22] The staff_member_required Decorator: Django Documentation. 2024.
https://docs.djangoproject.com/en/5.0/ref/contrib/admin/#django.contrib.admin.views.decorators.staff_member_required

[23] Logging: Django Documentation. 2024.
<https://docs.djangoproject.com/en/4.2/topics/logging/>

[24] The Django admin site: Django Documentation. 2024.
<https://docs.djangoproject.com/en/5.0/ref/contrib/admin/>

[25] What is CRUD? 2024. <https://www.codecademy.com/article/what-is-crud>

[26] What is Breadcrumb & How Does It Ease Navigation? Explained with Examples. 2024. <https://vwo.com/blog/why-use-breadcrumbs/>

[27] Customize the Django Admin with Python. 2024.
<https://realpython.com/customize-django-admin-python/>

[28] What is the purpose of the Django Secret Key? 2024.
<https://clouddevs.com/django/secret-key/>

[29] Protect Your Sensitive Data: A Guide to .env Files in Django. 2023.
<https://dev.to/defidelity/protect-your-sensitive-data-a-guide-to-env-files-in-django-499e>

[30] Install and configure Apache. 2024. <https://ubuntu.com/tutorials/install-and-configure-apache>

[31] How to Install Django with Apache on Ubuntu 22.04. 2022.
<https://www.linuxtuto.com/how-to-install-django-with-apache-on-ubuntu-22-04/>

Приложения

Приложение 1. Сравнительный анализ аналогов продукта

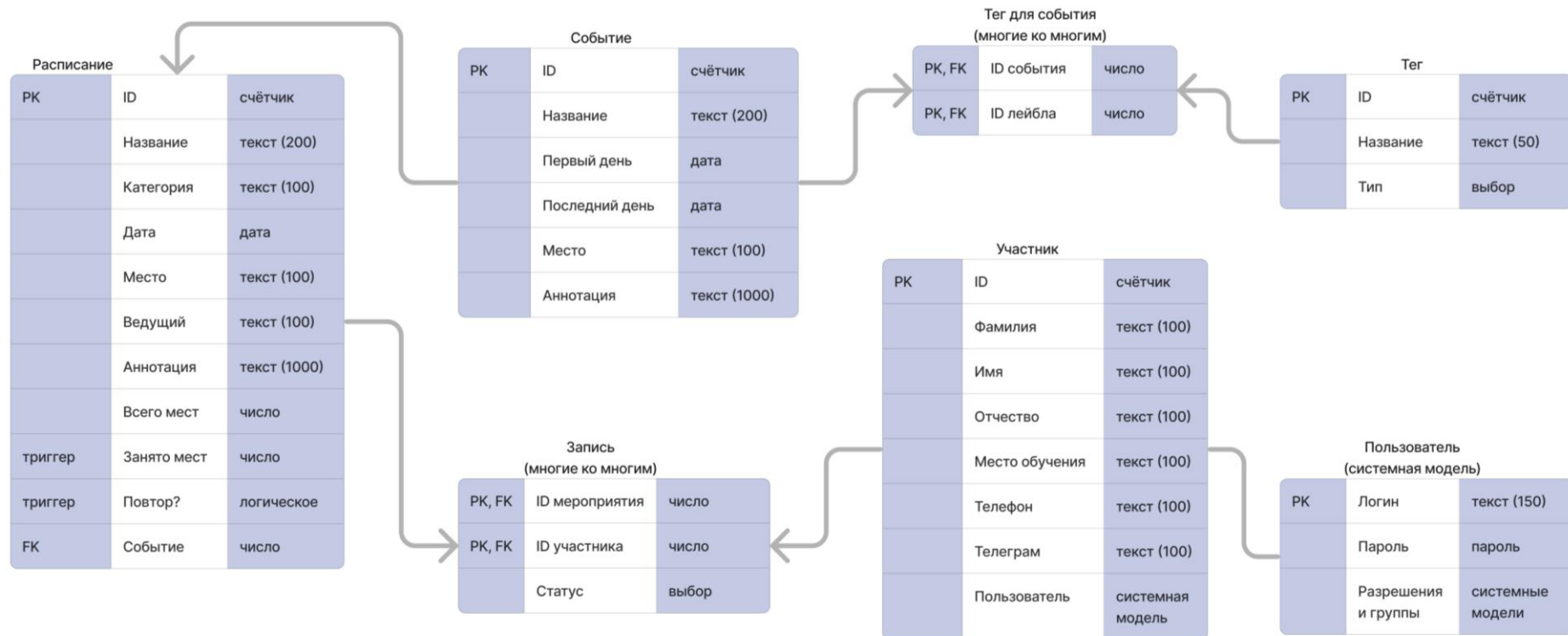
Продукт		Whova	Eventbrite	Wild Apricot	Bizzabo	Google Forms	Timepad	Планируемый продукт
Мероприятия	Большие (100+)	+	+	затратно	+	+	+	+
	Малые (99-)	затратно	+	+	+	+	+	+
	Регулярные	+	+	+	+	+	+	+
	Нерегулярные	+	+	затратно	+	+	+	+
	С ограничением мест	+	+	+	+	требуется разработки	+	+
Другой функционал	Расписание для пользователя	+	+	+	+	требуется разработки	нет	+
	Аналитика для организатора	+	+	+	+	+	+	+
	Рекламная рассылка	+	+	+	+	+	+	+
Стоимость использования		по билету	по билету	подписка	подписка	без комиссии	по билету	без комиссии
Платформа	Веб	нет	+	+	нет	+	+	+
	Десктоп	+	+	+	+	нет	нет	нет
	Мобильное приложение	+	+	+	+	нет	нет	нет

Приложение 2. Хранилище исходного кода

<https://github.com/yualapshina/registration-system-coursework>



Приложение 3. Архитектура базы данных



Приложение 4. Страница регистрации

[войти](#) [зарегистрироваться](#) [обратная связь](#) [справка](#)

регистрация

электронная почта

Введи твой адрес электронной почты

На введённый тобой адрес будет направлен пароль, который ты сможешь изменить после входа в профиль

☐ Я даю согласие на обработку персональных данных

☐ Я даю согласие на фотографирование

создать аккаунт

Приложение 5. Страница авторизации

[войти](#) [зарегистрироваться](#) [обратная связь](#) [справка](#)

авторизация

Только что зарегистрировался? Тогда ищи пароль на своей электронной почте – мы уже отправили его!

электронная почта

пароль

[войти](#) [не помню пароль](#)

Приложение 6. Страница профиля

[профиль](#) [моё расписание](#) [мероприятия](#) [обратная связь](#) [справка](#) [выйти](#)

данные для входа:

электронная почта

старый пароль

новый пароль

новый пароль (снова)

Пароль должен содержать как минимум 8 символов, одну заглавную букву и одну цифру

После изменения пароля потребуется заново зайти в профиль

личная информация:

фамилия

имя

отчество

место обучения

телефон

телеграм

Приложение 7. Страница мероприятий

профиль

моё расписание

мероприятия

обратная связь

справка

выйти

Аудитория

☐ Студентам

☐ Для всех желающих

Направление

☒ Развлечения

☐ IT

☐ Общая эрудиция

фильтровать

сбросить все фильтры

предстоящие мероприятия

Выпускной-2024

Для всех желающих

Развлечения

парк "Швейцария"

29 июня 2024 г. - 30 июня 2024 г.

хочу сюда

Приложение 8. Страница уточнения расписания

[профиль](#) [моё расписание](#) [мероприятия](#) [обратная связь](#) [справка](#) [ВЫЙТИ](#)

выпускной-2024: уточнение расписания

Будь внимателен! Некоторые мероприятия идут одновременно – выбрать можно только одно.
А еще, некоторые из них могут повторяться в разное время.

29 июня 2024 г.

Утро

☒ Репетиция

Место: центральная площадь Ведущий: Артём Нефёдов Свободных мест: 19/20

30 июня 2024 г.

10:00 - 15:00

☐ Торжественная часть

Место: центральная площадь Ведущий: Артём Нефёдов Свободных мест: не ограничено

15:00 - 18:00

☐ Вечеринка

Приложение 9. Список мероприятий участника

[профиль](#) [моё расписание](#) [мероприятия](#) [обратная связь](#) [справка](#) [выйти](#)

твое расписание

Выпускной-2024

29 июня 2024 г.

Утро	Репетиция	QR
парк "Швейцария", центральная площадь	Артём Нефёдов	Еще не посещено

30 июня 2024 г.

10:00 - 15:00	Торжественная часть	QR
парк "Швейцария", центральная площадь	Артём Нефёдов	Еще не посещено
15:00 - 18:00	Вечеринка	QR
парк "Швейцария", центральная площадь	Артём Нефёдов	Еще не посещено

[скачать](#) [изменить](#) [удалить](#)

Регистрация успешно обновлена

Приложение 10. Генерация расписаний

Выпускной-2024

29.06.2024

Утро: Репетиция, центральная площадь

30.06.2024

10:00 - 15:00: Торжественная часть, центральная площадь

15:00 - 18:00: Вечеринка, центральная площадь

Приложение 11. Генерация сертификатов

Национальный исследовательский университет
«Высшая школа экономики»
Нижний Новгород

СЕРТИФИКАТ
подтверждает, что

Лапшина Юлия Алексеевна


посетил(а) мероприятие


**Студенческий чемпионат России по
интеллектуальным играм**

с 03.05.2024 по 05.05.2024

Активности, посещенные
в рамках мероприятия:
Основная дисциплина, туры 1-3
Основная дисциплина, туры 4-6

должность важного человека
Очень Важный Человек

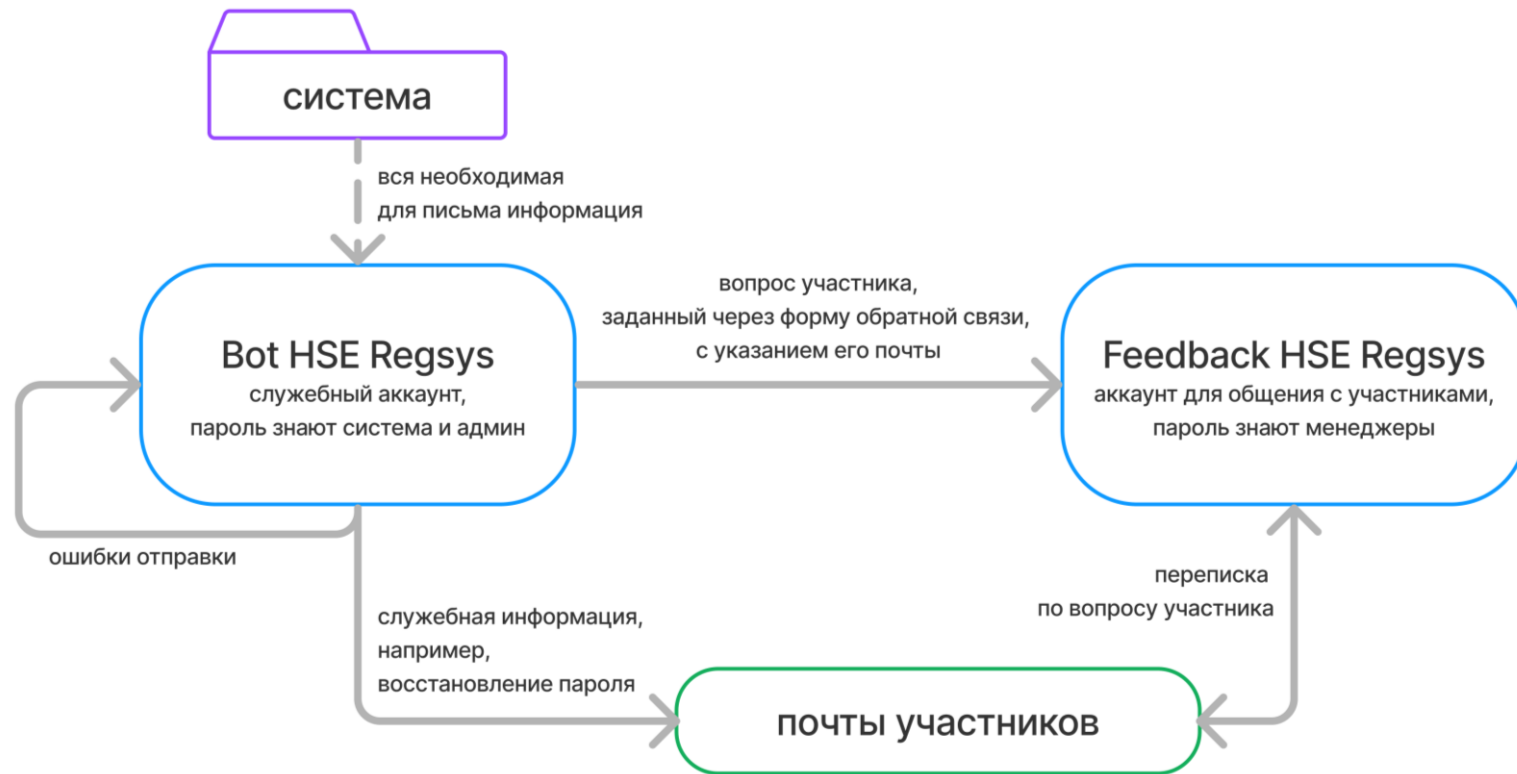




Приложение 12. log_info.log

INFO Watching for file changes with StatReloader
INFO "GET /mylist/ HTTP/1.1" 200 5643
INFO "GET /static/regsys/style.css HTTP/1.1" 304 0
INFO "GET /static/regsys/fonts.css HTTP/1.1" 304 0
INFO "GET /static/regsys/elements.css HTTP/1.1" 304 0
INFO "GET /static/regsys/colors.css HTTP/1.1" 304 0
INFO "POST /dispatcher/ HTTP/1.1" 302 0
INFO "GET /feedback/ HTTP/1.1" 200 2723
INFO "POST /admin/regsys/label/ HTTP/1.1" 200 9706
INFO "GET /static/admin/js/cancel.js HTTP/1.1" 200 884
INFO "POST /admin/regsys/label/ HTTP/1.1" 302 0
INFO "GET /admin/regsys/label/ HTTP/1.1" 200 14608
INFO "GET /admin/jsi18n/ HTTP/1.1" 200 16066
INFO "POST /dispatcher/ HTTP/1.1" 302 0
INFO "GET /feedback/ HTTP/1.1" 200 2864

Приложение 13. Схема обмена электронной почтой



Приложение 14. Почтовый диалог с пользователем

Размеры полей не удовлетворяют требованиям моих персональных данных.



Входящие x



noreply.hse.regsys@gmail.com

кому: мне ▾

18:01 (3 часа назад)



Премногоуважаемые разработчики сией прекрасной формы. К сожалению, я обнаружил, что при вводе моих личных данных на Вашем сайте поле "Место обучения" загадочно отреагировало на "Институт международных отношений и мировой истории Нижегородского государственного университета имени Н. И. Лобачевского", выдав некую странную и непонятную ошибку. Можно ли с этим что-то сделать?



Feedback HSE RegSys <feedback.hse.regsys@gmail.com>

кому: dm1try.y4sh1n ▾

21:36 (3 минуты назад)



Добрый день!

Большое спасибо, что поставили нас в известность! К сожалению, нам ранее не встречались студенты ИМОМИ ННГУ, которые так уважают свою альма-матер. Но мы учли ваше замечание и расширили поля данных, а также доработали форму так, чтобы она мягко сообщала пользователю о существующих ограничениях. Надеемся, что это улучшит впечатления от использования, и спасибо вам, что приняли участие в бета-тестировании нашего приложения!

--

Без негатива,
команда разработки HSE RegSys

Приложение 15. Администрирование – страница просмотра

Администрирование Django

ДОБРО ПОЖАЛОВАТЬ, **ЮЛИЯ**. [СПРАВКА](#) / [ОТКРЫТЬ САЙТ](#) / [ИЗМЕНИТЬ ПАРОЛЬ](#) / [ВЫЙТИ](#)

Начало > Система регистрации > Элементы расписания

Начните печатать для фильтрации...

ПОЛЬЗОВАТЕЛИ И ГРУППЫ

Группы + Добавить

Пользователи + Добавить

СИСТЕМА РЕГИСТРАЦИИ

Записи + Добавить

События + Добавить

Теги + Добавить

Теги событий + Добавить

Участники + Добавить

Элементы расписания + Добавить

Выберите Элемент расписания для изменения

🔍

Найти

Действие:

▼

Выполнить

 Выбрано 0 из 5

<input type="checkbox"/>	НАЗВАНИЕ	СОБЫТИЕ	КАТЕГОРИЯ	ДАТА	МЕСТО	ВЕДУ
<input type="checkbox"/>	Вечеринка	Выпускной-2024	15:00 - 18:00	30 июня 2024 г.	центральная площадь	Артём Нефё
<input type="checkbox"/>	Торжественная часть	Выпускной-2024	10:00 - 15:00	30 июня 2024 г.	центральная площадь	Артём Нефё
<input type="checkbox"/>	Репетиция	Выпускной-2024	Утро	29 июня 2024 г.	центральная площадь	Артём Нефё
<input type="checkbox"/>	Основная дисциплина, туры 4-6	Студенческий чемпионат России по интеллектуальным играм	12:00-15:00	5 мая 2024 г.	главный зал	Илья Козы
<input type="checkbox"/>	Основная дисциплина, туры 1-3	Студенческий чемпионат России по интеллектуальным играм	12:00-15:00	4 мая 2024 г.	главный зал	Илья Козы

◀

▶

5 Элементы расписания

ДОБАВИТЬ ЭЛЕМЕНТ РАСПИСАНИЯ +

ФИЛЬТР

👁 Показывать счётчики

↓ Событие

Все

Студенческий чемпионат России по интеллектуальным играм

Выпускной-2024

↓ Категория

Все

10:00 - 15:00

12:00-15:00

15:00 - 18:00

Утро

↓ Дата

Любая дата

Сегодня

Последние 7 дней

Этот месяц

Этот год

↓ Повтор?

Приложение 16. Администрирование – страница изменения

Администрирование Django

ДОБРО ПОЖАЛОВАТЬ, **ЮЛИЯ**. СПРАВКА / ОТКРЫТЬ САЙТ / ИЗМЕНИТЬ ПАРОЛЬ / ВЫЙТИ

Начало > Система регистрации > Элементы расписания > Торжественная часть

Начните печатать для фильтрации...

ПОЛЬЗОВАТЕЛИ И ГРУППЫ

Группы + Добавить

Пользователи + Добавить

СИСТЕМА РЕГИСТРАЦИИ

Записи + Добавить

События + Добавить

Теги + Добавить

Теги событий + Добавить

Участники + Добавить

Элементы расписания + Добавить

Изменить Элемент расписания

ТОРЖЕСТВЕННАЯ ЧАСТЬ

ИСТОРИЯ

Название: Торжественная часть

Категория: 10:00 - 15:00
* в категорию входит время проведения мероприятия (11.00-12.00) или его время и тип (14.00-15.00, обед); для корректности отображения расписания важно вводить время в одном формате

Дата: 30.06.2024 Сегодня

Место: центральная площадь

Ведущий: Артём Нефёдов

Аннотация: Каждый из выпускников получит свою мин

Событие: Выпускной-2024

Всего мест: -1

СОХРАНИТЬ

Сохранить и добавить другой объект

Сохранить и продолжить редактирование

Удалить

Приложение 17. Система регистрации в общем доступе

<https://hse-reg-sys.dns-dynamic.net/>



Приложение 18. apache/sites-available/django.conf

<VirtualHost *:80>

ServerAdmin admin@hse-reg-sys.dns-dynamic.net

ServerName hse-reg-sys.dns-dynamic.net

ServerAlias www.hse-reg-sys.dns-dynamic.net

DocumentRoot /var/www/reg-sys-coursework/

ErrorLog \${APACHE_LOG_DIR}/hse-reg-sys.dns-dynamic.net_error.log

CustomLog \${APACHE_LOG_DIR}/hse-reg-sys.dns-dynamic.net_access.log

comb>

Alias /static /var/www/reg-sys-coursework/static

<Directory /var/www/reg-sys-coursework/static>

Require all granted

</Directory>

<Directory /var/www/reg-sys-coursework/reg_sys_cw>

<Files wsgi.py>

Require all granted

</Files>

</Directory>

WSGIDaemonProcess reg_sys_cw python-path=/var/www/reg-sys-coursework py>

WSGIProcessGroup reg_sys_cw

WSGIScriptAlias / /var/www/reg-sys-coursework/reg_sys_cw/wsgi.py

RewriteEngine on

RewriteCond %{SERVER_NAME} =hse-reg-sys.dns-dynamic.net [OR]

RewriteCond %{SERVER_NAME} =www.hse-reg-sys.dns-dynamic.net

RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI}

[END,NE,R=permanent]

</VirtualHost>