Федеральное государственное автономное образовательное учреждение  
высшего образования   
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ   
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук   
НИУ ВШЭ-Нижний Новгород  
Образовательная программа «Программная инженерия»

**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:   
студентка 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 сравнительный анализ рынка подобных решений, 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

[Введение 4](#_Toc135302196)

[1. Проектирование системы 7](#_Toc135302197)

[1.1 Обзор существующих аналогов 7](#_Toc135302198)

[2. Средства и технологии разработки 10](#_Toc135302199)

[2.1 Выбор фреймворка 10](#_Toc135302200)

[2.2 Выбор СУБД 10](#_Toc135302201)

[3. Описание программного продукта 12](#_Toc135302202)

[3.1 Общая файловая структура 12](#_Toc135302203)

[3.2 Структура базы данных 12](#_Toc135302204)

[3.3 Запросы и передача данных 14](#_Toc135302205)

[3.4 Панель администрирования 16](#_Toc135302206)

[Заключение 18](#_Toc135302207)

[Источники 19](#_Toc135302208)

[Приложения 21](#_Toc135302209)

[Приложение 1. Сравнительный анализ аналогов продукта 21](#_Toc135302210)

[Приложение 2. Архитектура базы данных 22](#_Toc135302211)

[Приложение 3. Панель администрирования – вид таблицы 23](#_Toc135302212)

[Приложение 4. Панель администрирования – функционал добавления элемента 24](#_Toc135302213)

# Введение

В настоящее время информатизация настолько глубоко проникла в повседневную жизнь общества, что даже люди, не связанные непосредственно со сферой информационных технологий, очень во многих аспектах жизни полагаются на компьютеризированные решения. Наличие всевозможных систем получения, хранения и обмена информацией делает мгновенно доступным то, на что вживую пришлось бы тратить заметно больше времени, и таким образом значительно упрощает и рационализирует жизненные процессы – как с точки зрения человека-клиента, желающего получить необходимую ему услугу без привязки к времени и месту, так и для предоставляющей эту услугу стороны, которой не приходится тратить ресурсы на обработку потоков информации вручную [1].

Одной из многочисленных задач, требующих подобного автоматизированного решения, является регистрация. Многие учреждения и сегодня, устраивая мероприятия, поручают сотрудникам работу по подсчёту участников, сбору и систематизации их данных, которая вполне может производиться автоматически. Такое решение сделало бы участие приятнее и для гостей, ведь намного удобнее в текущем ритме жизни иметь возможность регистрироваться на мероприятия через Интернет, сразу получая информацию о доступности мест и прочих организационных моментах. Необходимость в подобном решении возникла и у нижегородского кампуса Высшей школы экономики, где мы обучаемся. Университет регулярно проводит открытые мероприятия как для студентов, так и для широкой аудитории – заинтересованных в поступлении школьников, а также просто всех желающих; при этом единой системы регистрации у кампуса не существует, и по необходимости каждое подразделение ищет собственные способы сбора информации и привлечением сторонних инструментов, не всегда понятных и удобных. Именно в необходимости рационализировать этот процесс, чтобы упростить организационную работу факультетов и улучшить впечатления посетителей, и заключается актуальность нашей работы.

Таким образом, целью её стала разработка системы, которая могла бы использоваться как для регистрации 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 простота и наглядность в использовании.

Основными этапами, пройденными по мере выполнения нами работы, стали:

* сбор требований от руководства и их формализация в виде технического задания для удовлетворения существующей необходимости в продукте;
* обзор существующих на рынке систем аналогичного назначения и их сравнение с нашим предполагаемым продуктом;
* планирование и распределение ролей в проекте, принимающее во внимание предыдущий опыт работы и дальнейшее желание развиваться;
* выбор инструментов и технологий разработки в соответствии с поставленными целями;
* выстраивание архитектуры и пошаговая разработка обоих аспектов продукта (серверной и клиентской части);
* интеграция проделанной работы, тестирование и финализация прототипа системы;
* ретроспектива и анализ соответствия проделанной работы изначальным планам;
* подготовка к будущему внедрению продукта в реально используемые сервисы НИУ ВШЭ.

Что касается этой индивидуальной работы, она рассматривает обзор существующих на рынке продуктов подобной категории, а также подробности реализации серверной и алгоритмической части системы (бэкенд, backend). В ней будет проведён сравнительный анализ возможных характеристик схожих систем, обоснован выбор технологий и инструментов для разработки и описана основная структура кода и программных компонентов.

# 1. Проектирование системы

## 1.1 Обзор существующих аналогов

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

Большую часть критериев составили 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 описать разницу следующим образом: тогда как публичная лекция или другое общедоступное мероприятие, без сомнения, может считаться и маленьким мероприятием, локальные вечеринки и праздники, на которых подразумевается, что гости более-менее знакомы друг с другом и составляют единую компанию, не должны считаться большими.
* 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 [2]:

* 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, для обеих сторон – возможность организации постоянной рекламной рассылки.
* Оценка стоимости использования сервиса, как для организатора, так и для посетителя. Три основных паттерна, выделившиеся при исследовании: 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 [3]. 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 [4].

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 [5]. Timepad, к сожалению, такой возможности не предоставляет.

Что касается планируемого для нашей собственной 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. Исходя из этих критериев, его логично отнести к группе инструментов, упомянутых последними, – брать пример с имеющихся сервисов любительского типа, реализуя в первую очередь самые необходимые для работы функции, но оставлять, естественно, возможность масштабирования продукта до полноценного коммерческого бренда.

# 2. Средства и технологии разработки

## 2.1 Выбор фреймворка

Переходя непосредственно к стеку 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 [6].

## 2.2 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 [7].

# 3. Описание программного продукта

## 3.1 Общая файловая структура

Исходный код регистрационной системы, вместе с пошаговой историей работы и её планирования (посредством GitHub Issues): [8]

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

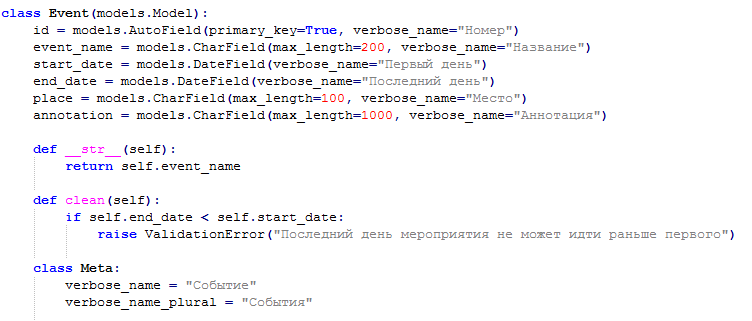
Наглядно изображённая схематически структура сущностей и отношений в используемой базе данных размещена в Приложении 2.

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 [9], 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:

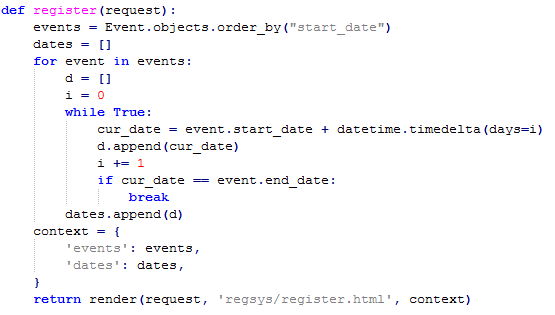
* реализована валидация, сопоставляющая **Event.start\_date** и **Event.end\_date** во избежание неверно указанных дат события;
* подобной же валидацией проверяется, что **Timetable.date** входит в рамки своего «родительского» события;
* с использованием механизма Django Сигналов [10] устроено автоматическое пересчитывание свободных мест при добавлении или удалении **Записи**.

Реализация таблицы посредством модели

Все вносимые в базу данных изменения контролируются фреймворком в директории *migrations* [11].

## 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).



Реализация представления

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.

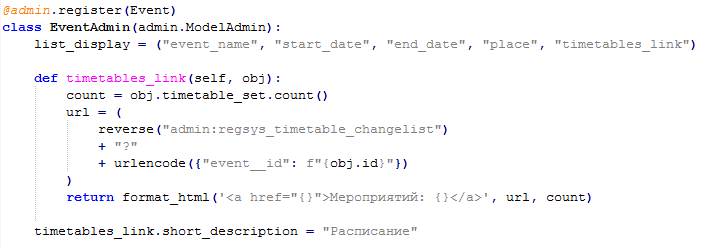
Реализовано также четвёртое представление, **download**, которое отличается от остальных форматом возвращаемого объекта. Для него не предусмотрено визуального шаблона – он возвращает сгенерированный файл. Эта функция отвечает за составление таблицы расписания, которая доступна пользователю для скачивания 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 [12].

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. Интерфейс инструмента представлен в Приложениях 3 и 4.

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 [13].

Декорирование таблицы в панели администрирования

# Заключение

Электронные системы сбора и обработки данных сегодня являются неотъемлемой частью нашей жизни, ускоряя и упрощая её в разы. Поучаствовать в разработке одной из них, особенно рассчитывая на то, что она получит шанс внедрения в сервис реальной организации, стало для меня очень интересной работой, принёсшей значительное количество опыта – не только в непосредственной разработке, но и в проектировании и анализе задач, а также в организации совместной работы и гибкости нахождения решений. Проделанную работу я оцениваю по большей части положительно и намерена развивать её дальше, дав «боевое» применение этому прикладному продукту. Считаю, что данная реализация регистрационной системы вполне удовлетворяет всем поставленным задачам 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] Горбунова О.Н. Информатизация общества и формирование трудового ресурса: проблемы, пути решения. Социально-экономические явления и процессы. 2012.

[2] 8 Popular Event Registration Software for Conferences and Events. 2023. <https://whova.com/blog/event-registration-software-price-comparison/>

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

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

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

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

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

[8] yualapshina/registration-system-coursework. 2023.

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

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

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

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

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

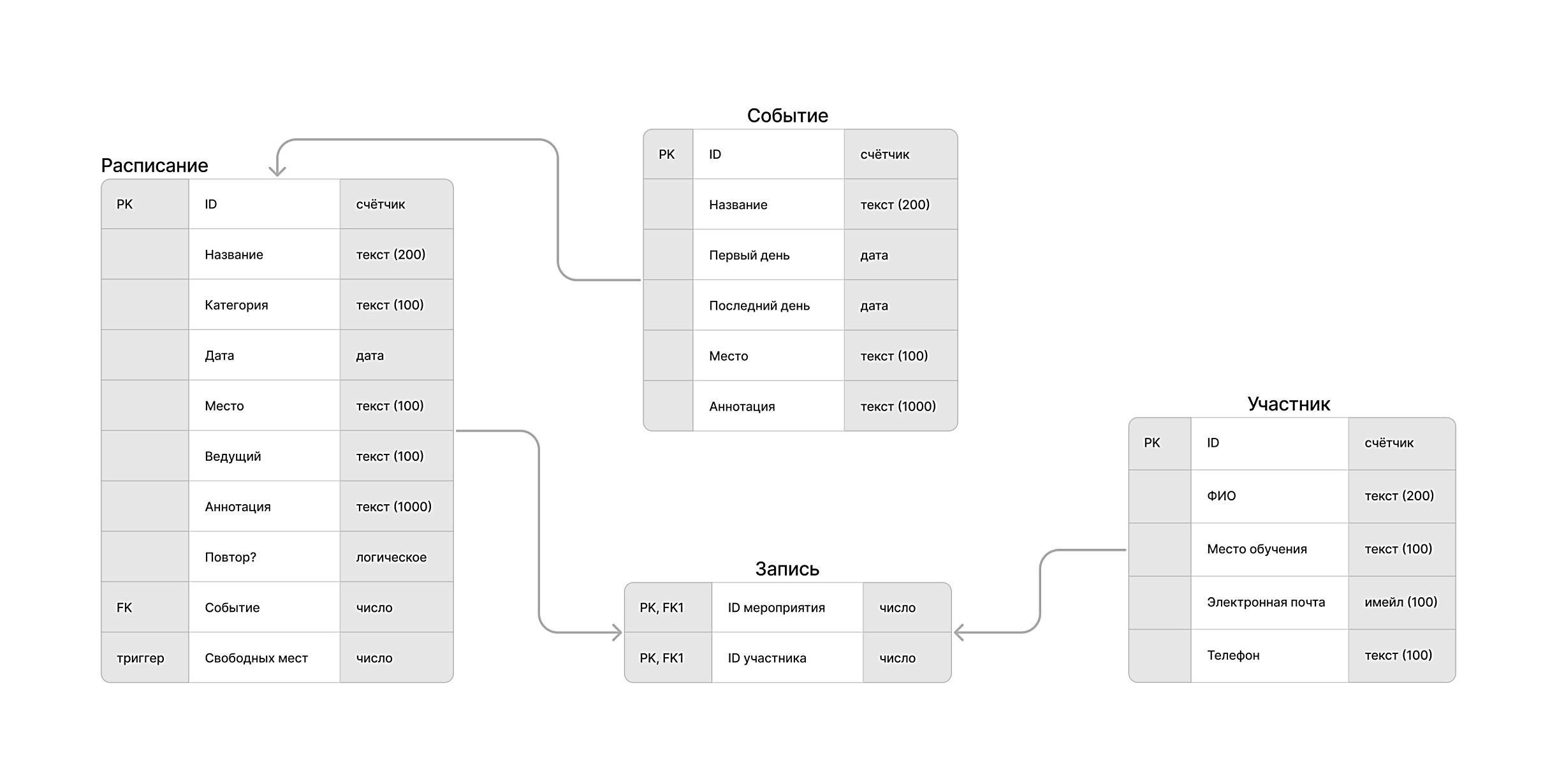
[13] Customize the Django Admin With Python. 2023. <https://realpython.com/customize-django-admin-python/#providing-links-to-other-object-pages>

# Приложения

## Приложение 1. Сравнительный анализ аналогов продукта

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

## Приложение 2. Архитектура базы данных



## Приложение 3. Панель администрирования – вид таблицы

## Приложение 4. Панель администрирования – функционал добавления элемента