**2 Проектирование задачи**

**2.1 Моделирование проекта**

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (действующим лицом, актантом) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. Данная диаграмма изображена на рисунке 2.1.

Диаграмма базы данных – это наглядное представление таблиц и связей между ними.

Сущность (таблица, отношение) — это представление набора реальных или абстрактных объектов (людей, вещей, мест, событий, идей, комбинаций и т. д.), которые можно выделить в одну группу, потому что они имеют одинаковые характеристики и могут принимать участие в похожих связях. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность в модели изображается в виде прямоугольника с наименованием.

Можно сказать, что Сущности представляют собой множество реальных или абстрактных вещей (людей, объектов, событий, идей и т. д.), которые имеют общие атрибуты или характеристики.

Экземпляр сущности (запись, кортеж) - это конкретный представитель данной сущности.

Атрибут сущности (поле, домен) — это именованная характеристика, являющаяся некоторым свойством сущности.

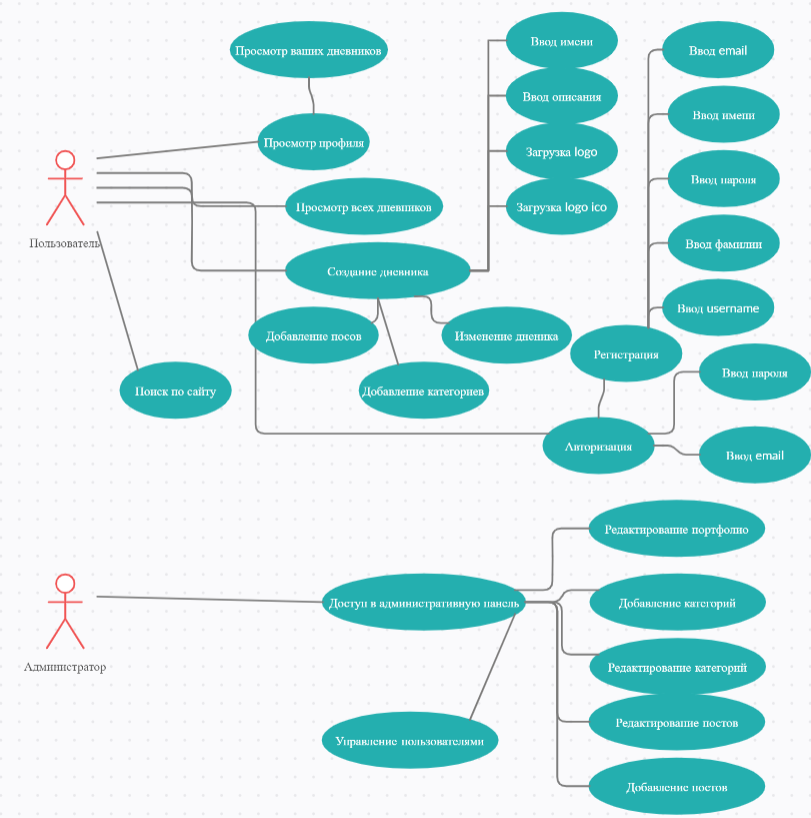
****

Рисунок 2.1 – Диаграмма вариантов использования

Источник: собственная разработка

Связь – это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Каждая связь может иметь один из следующих типов связи:

один-к-одному;

многое-ко-многим;

один-ко-многим.

Связь типа один-к-одному означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа многое-ко-многим означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Тип связи много-ко-многим является временным типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности.

Связь типа один-ко-многим означает, чтоодин экземпляр первой сущности(левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны «один») называется родительской, правая (со стороны «много») — дочерней.

Диаграмма базы данных представлена на рисунке 2.2.

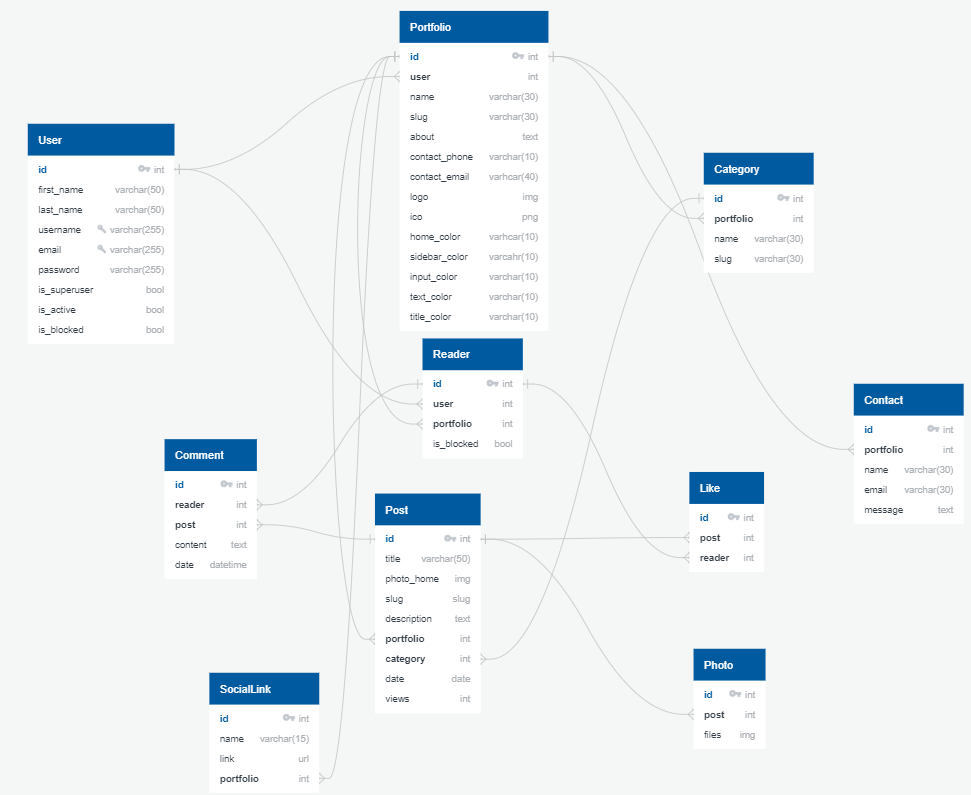


Рисунок 2.2 – Диаграмма базы данных

Источник: собственная разработка

Для моделирования процесса выполнения операций используются диаграммы деятельности. Именно на диаграмме деятельности представлены переходы потока управления от одной деятельности к другой. Это, по сути, разновидность диаграммы состояний, где все или большая часть состояний являются некоторыми деятельностями, а все или большая часть переходов срабатывают при завершении определенной деятельности и позволяют перейти к выполнению следующей. Как мы уже говорили (повторение - мать учения), диаграмма деятельности может быть присоединена к любому элементу модели, имеющему динамическое поведение. Кстати, исходя из вышесказанного, логичнее говорить не "диаграмма деятельности", а "диаграмма деятельностей" - во множественном числе. А еще мы предполагаем, что читатель понимает смысл понятий "деятельность", "переход" и "объект". Об объектах как об экземплярах классов мы уже говорили ранее. Понятия же деятельности (activity) как протяженного во времени составного (неатомарного) вычисления (действия, action) и перехода как передачи контроля, надеемся, понятны интуитивно, без дополнительных объяснений.

Диаграммы деятельности позволяют моделировать сложный жизненный цикл объекта, с переходами из одного состояния (деятельности) в другое. Но этот вид диаграмм может быть использован и для описания динамики совокупности объектов. Они применимы и для детализации некоторой конкретной операции, причем, как мы увидим далее, предоставляют для этого больше возможностей, чем "классическая" блок-схема. Диаграммы деятельности описывают переход от одной деятельности к другой, в отличие от диаграмм взаимодействия, где акцент делается на переходах потока управления от объекта к объекту. Диаграмма изображена на рисунке 2.3.

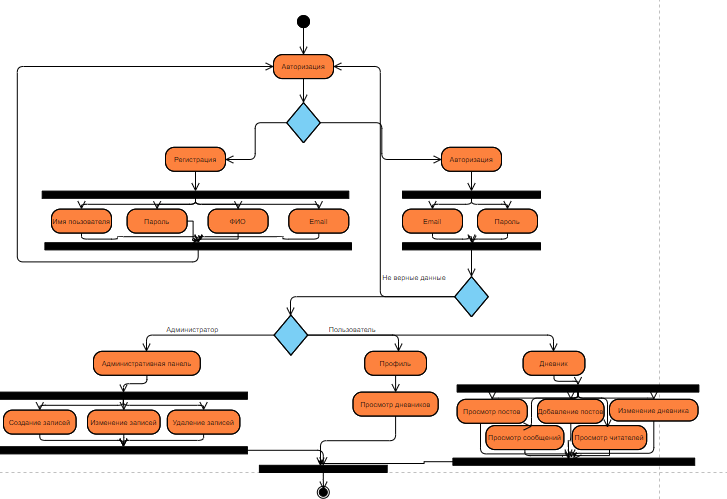


Рисунок 2.3 – Диаграмма деятельности

Источник: собственная разработка

**2.2 Описание системы меню**

При переходе на главную страницу сайта, расположены следующие кнопки: «Главная», «Авторизация», «Регистрация», «Список дневник». Ниже представлены доступные шаблоны дневников.

При переходе на вкладку «Авторизация» или «Регистрация» можно пройти авторизацию или регистрацию на сайте.

На вкладке «Список дневников» показаны все существующие дневники и краткая информация о них.

После авторизации будет доступна вкладка «Профиль», где можно посмотреть ваши дневники.

**2.3 Выбор и обоснование среды разработки**

Для решения задач был выбран следующий язык программирования и фреймворк соответственно: Python, Django.

Python – это широко используемый язык программирования высокого уровня для программирования общего назначения. Python обладает динамической системой типов и автоматическим управлением памятью и поддерживает множество парадигм программирования, включая объектно-ориентированную, императивную функциональное программирование и процедурные стили.

С ним легко работать, что сокращает время разработки. Написанный в удобочитаемом формате, Python делает процесс разработки программного обеспечения быстрым, удобным и максимально упрощенным. Если разработчик имеет базовые знания любого другого языка программирования, то может легко изучить Python и внедрить его в свои проекты.

По сравнению с другими языками, Python в 5-10 раз быстрее по времени разработки, однако медленный при выполнении программ. Обеспечивает расширенные возможности управления процессами и объектно-ориентированный дизайн, помогая как в скорости, так и в производительности. Упрощенный контекст и удобные для пользователя структуры данных позволяют разработчикам легко читать и писать. Сокращенные сроки, связанные с Python, также означают меньшие затраты на разработку для компаний-разработчиков Python и их клиентов

Большое интернет-сообщество поддерживает Python. Скорее всего, если разработчик столкнется с проблемой или обнаружит ошибку в Python, другой разработчик уже найдет решение этой проблемы. Python поддерживается большим онлайн-сообществом сторонников и других программистов, которые постоянно улучшают его функциональные возможности. Кроме того, при его использовании легко получить быструю поддержку по сложным вопросам.

Django – фреймворк для веб-приложений на языке Python. Один из основных принципов фреймворка – DRY (don't repeat yourself). Веб-системы на Django строятся из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из заметных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Также, в отличие от многих других фреймворков, обработчики URL в Django конфигурируются явно (при помощи регулярных выражений), а не автоматически задаются из структуры контроллеров.

Django проектировался для работы под управлением Apache (с модулем mod\_python) и с использованием PostgreSQL в качестве базы данных. В настоящее время, помимо PostgreSQL, Django может работать с другими СУБД: MySQL (MariaDB), SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle. Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (Templates). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV).

Первоначально разработка Django велась для обеспечения более удобной работы с новостными ресурсами, что достаточно сильно отразилось на архитектуре: фреймворк предоставляет ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами (с пообъектным назначением прав).

PostgreSQL – это система управления базами данных с открытым исходным кодом корпоративного класса. Он поддерживает как SQL для реляционных, так и JSON для нереляционных запросов. Он поддерживается опытным сообществом разработчиков, которые внесли огромный вклад в создание высоконадежной системы СУБД.

PostgreSQL поддерживает расширенные типы данных и улучшенную оптимизацию производительности, функции, доступные только в дорогой коммерческой базе данных, такой как Oracle и SQL Server.

PostgreSQL предлагает мощную ГИС, которая называется «PostGIS». Это расширение предоставляет сотни функций для обработки геометрических данных в разных форматах. PostGIS полностью соответствует стандартам. Более того, используя как QGIS, так и GeoServer, сообщество Open Source предоставляет самый простой способ обработки геоданных.