Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия
« _____» _____ 2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – «Информатика и вычислительная техника»
код – наименование направления

Веб–приложение по созданию фотоальбома–отчета о событии
тема

Руководитель	_____ подпись, дата	канд. техн. наук, доцент должность, ученая степень	А. И. Постников инициалы, фамилия
Консультант	_____ подпись, дата	ст. преподаватель должность, ученая степень	О. В. Шмелев инициалы, фамилия
Выпускник	_____ подпись, дата		Е. В. Абросимов инициалы, фамилия
Нормоконтролер	_____ подпись, дата	канд. техн. наук, доцент должность, ученая степень	А. И. Постников инициалы, фамилия

Красноярск 2020

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия
« _____» _____ 2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме бакалаврской работы
бакалаврской работы, дипломного проекта, дипломной работы, магистерской диссертации

Студенту Абросимову Евгению Викторовичу
фамилия, имя, отчество

Группа ВКИ15-06Б Направление (специальность) 09.03.01
Номер Код

Информатика и вычислительная техника
наименование

Тема выпускной квалификационной работы Веб–приложение по созданию фотоальбома–отчета о событии

Утверждена приказом по университету № _____ от _____
Руководитель ВКР А. И. Постников, канд. техн. наук, доцент, ИКИТ СФУ
инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР Задание на ВКР

Перечень разделов ВКР Задание на ВКР, анализ задания, проектирование и реализация.

Перечень графического материала Презентация, выполненная с помощью Microsoft PowerPoint 2013.

Руководитель ВКР _____ А. И. Постников
подпись инициалы и фамилия

Задание принял к исполнению _____ Е. В. Абросимов
подпись, инициалы и фамилия студента

« ____ » _____2020г.

РЕФЕРАТ

Пояснительная записка ВКР «Веб–приложение по созданию фотоальбома – отчета о событии» / ИКИТ СФУ, Красноярск, 2020. Содержит 33 страницы, 20 рисунков, 4 таблицы, 8 источников.

ВЕБ-ПРИЛОЖЕНИЕ, ФОТОАЛЬБОМ, ОТЧЕТ, СОБЫТИЕ, DJANGO, PYTHON, БАЗАДАННЫХ.

Выпускная квалификационная работа посвящена проектированию и разработке веб–приложения на языке Python. Реализована методология ICONIX фокусирующая свое внимание на фазе анализа и дизайна. Рассмотрены основные технологии разработки веб–приложений, за основу взят шаблон проектирования MVC.

Веб–приложение создано с помощью Python–фреймворка Django 2.0. HTM, CSS, JS фреймфорка Bootstrap 4.0.

Результатом данной работы является разработка кроссплатформенного веб–приложения, осуществляющего функцию визуализации и изменения EXIF–данных фотографий и галерей, работающее в автономном режиме.

Итогом будет являться кроссплатформенное веб–приложение, реализующее интерактивный инструмент подготовки фотоальбома–отчет о событии, составленного из фотографий, полученных из разных источников.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном обществе огромное внимание уделяется фотографиям. Мы фотографируем всё, всех и с завидной регулярностью. Фотография – это искусство уже соревнующиеся с живописью из-за доступности и красоты передачи. С каждым днем, все лучше, четче, живее становятся снимки. Каждый снимок – это кусочек жизни автора. Каждая фотография, это момент, запечатленный в вечности, и никакими словами не передать то, что можно увидеть на красивом снимке.

На сегодняшний день, очень многие люди пользуются возможностью фотографировать. Очень многие производители различных гаджетов предоставляют возможность запечатлеть интересные моменты. И чем больше развивается фотография как искусство, тем чаще возникают ситуации обилия фотографий, созданных разными авторами с разных ракурсов и с минимальной задержкой во времени.

Вспомните любые свадьбы, походы, экспедиции. Любые мероприятия, в которых участвуют несколько человек с фотоаппаратами. После подобных событий, люди часами сидят и перебирают фотографии разных авторов. Огромные объемы, разные ракурсы, разные временные промежутки и качество фотографий порой приводят пользователей в ужас. Как итог, на дисках хранятся отложенные на потом гигабайты похожих фотографий и чем дальше, тем их больше. В данной ситуации, приложение по обработке большого числа снимков может отказаться незаменимым.

Целью данной работы является разработка автономного веб-приложения реализующее интерактивный инструмент подготовки фотоальбома–отчет о событии, составленного из фотографий, полученных из разных источников. Приложения должно предоставлять пользователю возможность загружать несколько папок фотографий, сортировать по времени, визуально сравнивать две

фотографии, вносить изменения в метаданные фотографий. И создавать новый альбом из выбранных изображений.

ГЛАВА 1. АНАЛИЗ ЗАДАНИЯ НА ПРОЕКТИРОВАНИЕ

Необходимо разработать веб–приложение, позволяющее через браузер:

- выбрать несколько каталогов, содержащих исходные фотографии в формате JPEG от разных авторов, и отобразить их в виде thumbnail–линеек, масштабированных и синхронизированных по времени из EXIF–данных;
- применить к любым из загруженных фотографий неискажающее изменение ориентации;
- визуально сравнить две или более указанных фотографий в отдельном окне с возможностью изменения масштаба;
- задать сдвиг по времени между загруженными каталогами; масштабирование и синхронизация между линейками должны измениться соответственно;
- сохранить изменения в EXIF–данные фотографий; сформировать фотоальбом, составленный из выбранных фотографий из разных каталогов, с изменением размеров фотографий на заданный; аннотировать фотографии фотоальбома;
- сохранить фотографии фотоальбома и аннотации в указанном каталоге с использованием механизма шаблонов.

1.1 Основные понятия

Веб–приложение – это клиент-серверное приложение, динамический веб-сайт с интерактивным содержимым. Это означает, что пользователь может взаимодействовать с содержимым страниц, заполняя формы, нажимая кнопки, запрашивая отчеты.

Веб приложение разделяют на фронтенд, то что клиент видит в окне своего браузера и бэкенд – логика, процессы, взаимодействие с базами данных происходящая на стороне сервера. В большинстве случаев применяться паттерн проектирования MVC.

MVC(Model–View–Controller) расшифровывается как Модель-Представление-Контроллер. Это набор идей и программных решений кода, когда выделяются блоки, отвечающие за решение разных задач.

Модель – отвечает за данные, определяет структуру и содержит бизнес-логику приложения.

Представления – отвечает за взаимодействие с пользователем, отображение данных генерируется представлением или видом.

Контроллер – отвечает за связь между моделью и представлением. Определяет как сайт будет реагировать на действия пользователя.

В Django другие названия для разделения данных и MVC называют MTV(Model–Templates–View), где view выполняют роль контроллера, а templates роль view.

Обычно веб–приложение состоит, минимум из трех основных компонентов.

1.1.1 Клиентская часть

Клиентская часть веб–приложения – реализует пользовательский интерфейс, то, что пользователь видит в браузере, и может взаимодействовать с сервером формируя запросы и получая ответы от него. Веб-интерфейс отображается в браузере клиента. Пользователь производит взаимодействие с веб-приложением именно через браузер. Нажимая кнопки, устанавливая галочки и заполняя формы.

1.1.2 Серверная часть

Серверная часть веб–приложения – программная обеспечение на сервере, которая обрабатывает HTTP запросы, полученные с клиентской стороны, формирует и выдает ответ, например, в виде HTML страницы обратно в браузер пользователю. Ответ не всегда нужен, это могут быть и просто изменения внутри сервера или базы данных. Популярные языки бэкенда PHP, Python, Java, Ruby.

1.1.3 База данных

База данных – это именованный набор данных, хранящийся в структурированном виде и отражающий связи и отношение объектов предметной области. Вид хранения и управления данными определяются системой управления базами данных(СУБД). СУБД программное обеспечение, предназначенное для организации, ведение и доступ к базам данных.

1.2 Выбор средств разработки

Сегодня существует много языков программирования, средств и сред разработки. Все они удобны для той или иной ситуации, однако для написания веб–приложения используются следующие инструменты:

- Python
- Django
- Bootstrap
- PyCharm
- Webstorm
- SQLite

1.2.1 Язык программирования Python

Python – это высокоуровневый, интерпретируемый, язык программирования общего назначения. Поддерживает большинство парадигм. Основное направления языка, облегчить и ускорить время разработки. Обладает лаконичным синтаксисом и обладает широким выбором библиотек в различных сферах деятельности.

Плюсы:

- Кроссплатформенность;
- Возможность подключить библиотеки написанные на C;
- Открытая разработка, простоту обучения на начальном этапе;
- Особенности синтаксиса позволяют писать хорошо читаемый код;
- Динамическая типизация.

Минусы:

- Скорость выполнения;

1.2.2 Django – фреймворк для веб–разработки на Python

Django – это популярный, мощный фреймворк для веб–приложений на языке Python. Использующий шаблон проектирования MVC. Один из основных принципов фреймворка – DRY (don't repeat yourself / не повторяйте себя). Тысячи сайтов созданы на Django. Этот фреймворк отлично подойдет для разработки веб-приложения.

Плюсы:

- Быстрота создания приложений;
- Множество дополнительных виджетов и функций;
- Масштабируемость;
- безопасность.

Минусы:

- Монолитность;
- все базируется на ORM Django;

- Шаблоны маршрутизации с указанием URL;
- компоненты развертываются совместно.

1.2.3 Реляционная система управления базами данных SQLite

SQLite – библиотека, встраиваемая в приложение, которое её использует. Будучи файловой БД, она предоставляет отличный набор инструментов для более простой (в сравнении с серверными БД) обработки любых видов данных.

Когда приложение использует SQLite, их связь производится с помощью функциональных и прямых вызовов файлов, содержащих данные (например, баз данных SQLite), а не какого–то интерфейса, что повышает скорость и производительность операций.[1]

Плюсы:

- Файловая структура, так как вся база данных хранится в одном файле, её легко переносить, не нужен сервер СУБД;
- Ссвободная лицензия;
- Высокая скорость;

Минусы:

- Отсутствие системы пользователей.

1.2.4 Интегрированная среда разработки PyCharm

PyCharm – популярная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит–тестов и поддерживает веб–разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.[2]

PyCharm – это кросс–платформенная среда разработки, которая совместима с основными операционными системами.

Плюсы:

- Понятный git;

- Простая организация проеков;
- Тесная интеграция с Django
- Быстрый.

Минусы:

- Цена

1.2.5 Фреймворк Bootstrap

Bootstrap – это фреймворк из трёх языков HTML/CSS/JS. Открытый и бесплатный набор инструментов для создания сайтов и веб–приложений. Включает в себя HTML и CSS–шаблоны оформления для кнопок, веб–форм, вкладок, меток, блоков навигации, каруселей и прочих компонентов веб–интерфейса, включая JavaScript–расширения.[3]

Появился в стенах компании Twitter и назывался «Twitter Bootstrap». Но из–за того что его захотели сделать всемирным пришлось отказаться от слова Twitter в названии.

Плюсы:

- быстрота верстки;
- адаптивность;
- популярность.

Минусы:

- перегруженность шаблонов.

1.2.6 Интегрированная среда разработки Webstorm

WebStorm – это интегрированная среда разработки на JavaScript, CSS & HTML так же, как и PyCharm от компании JetBrains. WebStorm обладает всеми достоинствами и возможностями современных редакторов и превосходит их.

1.2.7 Обоснование выбора средств разработки

Для выполнения данной работы со стороны бэкэнда был Python фреймворк Django вместе с базой данных SQLite, так как Python имеет встроенную поддержку SQLite базы данных и достаточно импортировать стандартную библиотеку. Показывают хорошее взаимодействие друг с другом и довольно просты в написании веб–приложения. Так как веб–приложение должно работать в автономном, монопольном и кроссплатформенном режиме, эти средства подходят лучше всего.

Фреймворк Bootstrap был выбран, как удобный и быстрый в написании инструментарий, обладающий большой библиотекой шаблонов, которая в свою очередь, позволит создавать дружественный пользователю графический интерфейс.

Среды разработки PyCharm и Webstorm являются коммерческими продуктами, и созданы под специализированные задачи, а именно PyCharm работу с Python, а Webstorm инструмент для фронтенд разработчиков. В бесплатном варианте PyCharm можно использовать в урезанном варианте, а Webstorm в течении 30 дней. В отличии от бесплатных Atom или Visual Studio Code, но имеющих более широкий спектр задач. Но разработчик PyCharm и Webstorm Jetbrains предоставляет академические лицензии для обучавшихся на сайте stepik.org, позволявшие в течении трех месяцев в полной мере пользоваться их продуктами. Такая лицензия была получена и срок в три месяца достаточен, для выполнения выпускной квалификационной работы.

1.3 Методология разработки

Существуют различный методологии, парадигмы и модели разработки программного обеспечения. В этой работе будет использоваться методология

ICONIX, так как она приближена к RAD (Rapid Application Development — быстрая разработка приложений) и является нечто средним между RUP (Rational Unified Process - рациональный унифицированный процесс) и XP (Extreme Programming – экстремальное программирование).

При разработке веб-приложения использовалась система контроля версий Git, репозиторий размещался на хостинге Github.

Git – это распределённая система управления версиями, которая позволяет разработчикам работать совместно и отслеживать изменения в файлах. Каждый раз, когда сохраняется проект, система запоминает, как выглядят файлы в этот момент и сохраняет ссылку на этот снимок.

1.4 Выводы по разделу

Целью раздела являлось изучение предметной области, в результате которой были сформулированы требование в разрабатываемому веб–приложению. Были выбраны средства разработки и полностью определен технологический стек:

- HTML, CSS – языки разметки;
- Bootstrap – для построения фронтенда;
- SQLite – База данных;
- Django – для построения бэкенда;
- Python – язык программирования.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ

Сегодня процесс создания сложных программных приложений невозможно представить без разделения на этапы жизненного цикла. Остановимся детально на процессе проектирования. В ходе проектирования архитектором или опытным программистом в начале нужно определить

первоначальные цели и область решаемых задач, затем собрать требования к веб-приложению и построить иерархическую структуру работ.

В основе методологии ICONIX лежат следующие приемы разработки:

1. Анализ требований – создаются модели прецедентов, модель пользовательского интерфейса и модель сущностей предметной области.
2. Предварительное проектирование – создается диаграмма пригодности, также дополняется модель прецедентов и модель сущностей предметной области.
3. Проектирование – создается диаграмма последовательности и создается диаграмма классов.
4. Реализация – создается исходный код, при этом возможно создание диаграммы развертывания и диаграммы компонентов, если это необходимо. [4]

В начале проанализируем информационную модель будущего веб-приложения. Составим модель прецедентов, модель первоначальной страницы и модель сущностей.

2.1 Модель прецедентов

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, актором) 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. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий. [5]

Веб-приложение предназначено для создания фотоальбома отчета о событии. Опишем процесс работы пользователя в модели прецедентов. Вначале пользователю нужно загрузить фотографии из нескольких источников, далее выбрать из них необходимые, путем визуального сравнения, далее сформировать из выбранных фотографий общий альбом, написать к нему аннотацию и выгрузить в указанную папку.

В разрабатываемом веб-приложении будет одна роль пользователя, работающего в автономном режиме, прохождение авторизации на данной итерации рассматривать не будем. Диаграмма прецедентов представлена на рисунке 1.

Рисунок 1 – Диаграмма прецедентов

Текстовое описание прецедентов:

Наименование прецедента: добавить папку.

Действующее лицо: Пользователь.

Цель: добавление папки с фотографиями.

Предусловия: пользователь зашел на главную страницу веб-приложения.

Главная последовательность:

a) Пользователь нажимает в начальном окне кнопку «Добавить папку»

b) Веб-приложение отображает страницу загрузки

Постусловие: Переход на страницу загрузки.

Наименование прецедента: загрузить.

Действующее лицо: Пользователь.

Цель: загрузка файлов фотографий.

Предусловия: пользователь нажал кнопку «Добавить папку».

Главная последовательность:

a) Пользователь заполняет название галереи и нажатием кнопки «Обзор» в позиции «Photo» открывает диалоговое окно выбора фотографий.

b) После выбора фотографий, нажатием кнопки «Загрузить»

c) Фотографий загружаются на сервер, пути к файлам сохраняются в базе данных.

d) На главной странице загруженные фотографии отображаются в виде горизонтальной галереи миниатюр с подписанным временем и галочками для выбора.

Постусловие: в главном окне отображается галерея миниатюр.

Наименование прецедента: сформировать альбом.

Действующее лицо: Пользователь.

Цель: сформировать из выбранных фотографий общий альбом, отсортированный по времени из EXIF данных.

Предусловия: пользователь добавил папки с фотографиями и выбрал нужные.

Главная последовательность:

a) Пользователь нажимает кнопку «Сформировать альбом».

b) Веб-приложение переходит на страницу альбома

c) Выгружает форму отображения выбранных фотографий.

Постусловие: в главном окне отображается галерея миниатюр из

выбранных фотографий и формой заполнения аннотации.

Наименование прецедента: сохранить альбом.

Действующее лицо: Пользователь.

Цель: сохранить из выбранных фотографий альбом с аннотацией.

Предусловия: пользователь сформировал альбом из выбранных

Главная последовательность:

a) Пользователь заполняет форму с аннотацией и нажимает кнопку «Сохранить».

b) Веб-приложение открывает диалоговое окно с выбором пути для сохранения.

Постусловие: Веб-приложение переводит в начальное окно и информирует о выгрузки файлов.

2.2 Модель пользовательского интерфейса

Прототип — это наглядная, схематичная модель пользовательского интерфейса веб- приложения. В него входят основные элементы дизайна и отображают его базовую структуру. По сути, это «черновик» интерфейса, созданный на основе ваших пожеланий о потребностях пользователя, без учета художественного оформления. Для создания прототипов использовалось ПО «BalsamiqMockups». Данный инструмент дает возможность построение каркасного изображения и создания прототипов графических интерфейсов.

Веб-приложение будет представлять из себя, «шапку» с логотипом и названием программы, центральную часть будут занимать thumbnail-линейки, которые можно добавлять нажатием кнопки в нижнем правом углу, так же там будет кнопка формирования альбома из выбранных фотографий, выбор осуществляется установкой флага по фотографии. В левой части увеличено показываться «начальные» фотографии, от указанного в EXIF-данных времени которых, будут строиться последующие изображения. Прототип сайта представлен на рисунке 2.

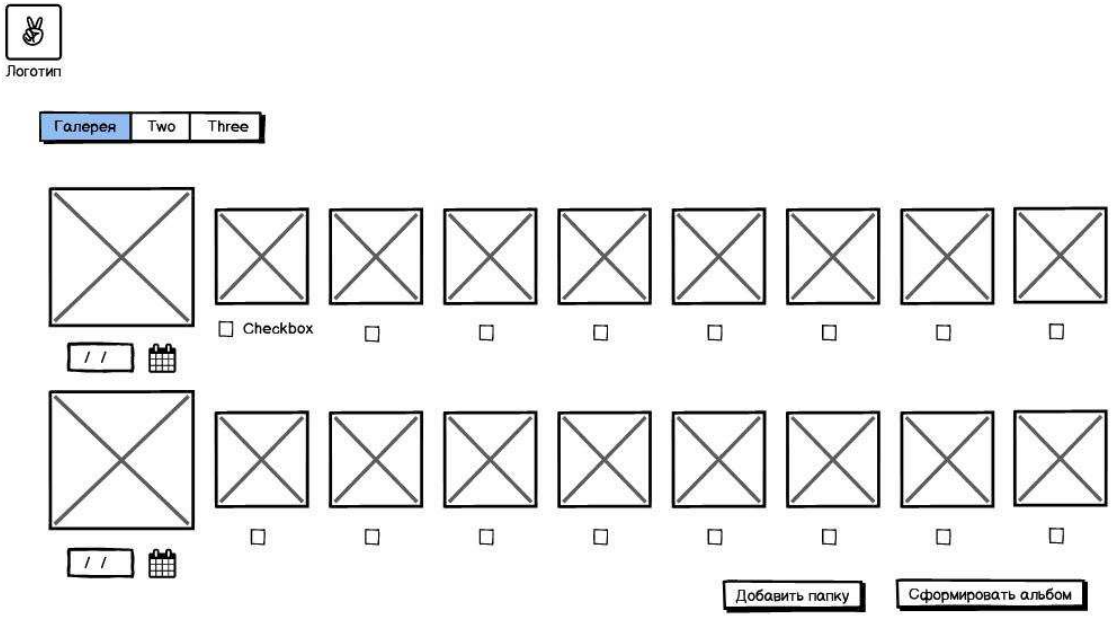

Рисунок 2 – Прототип пользовательского интерфейса.

2.3 Модель сущностей предметной области

Модель «сущность-связь» — это абстрактная модель, предназначенная для логического представления данных предметной области. Она предназначена для упрощения проектирования баз данных. Использует следующие основные понятия: сущности, взаимосвязи и атрибуты для представления свойств и взаимосвязей сущностей.

Сущность – это объект реального(физического) или концептуально(абстрактного) мира с одинаковыми свойствами, который можно однозначно идентифицировать. Каждая сущность обладает своим именем и набором атрибутов. Все сущности хранятся в базе данных.

Веб-приложение взаимодействует со следующими таблицами в базе данных db.sqlite3.

Модели-сущности веб-приложения

- Image – фотография
- Thumbnail – миниатюра

- Gallery – галерея из загруженных фотографий
- Album – общий альбом из нескольких галерей

Модели сущностей и связей разрабатываемого веб-приложения представлена на рисунке 3.

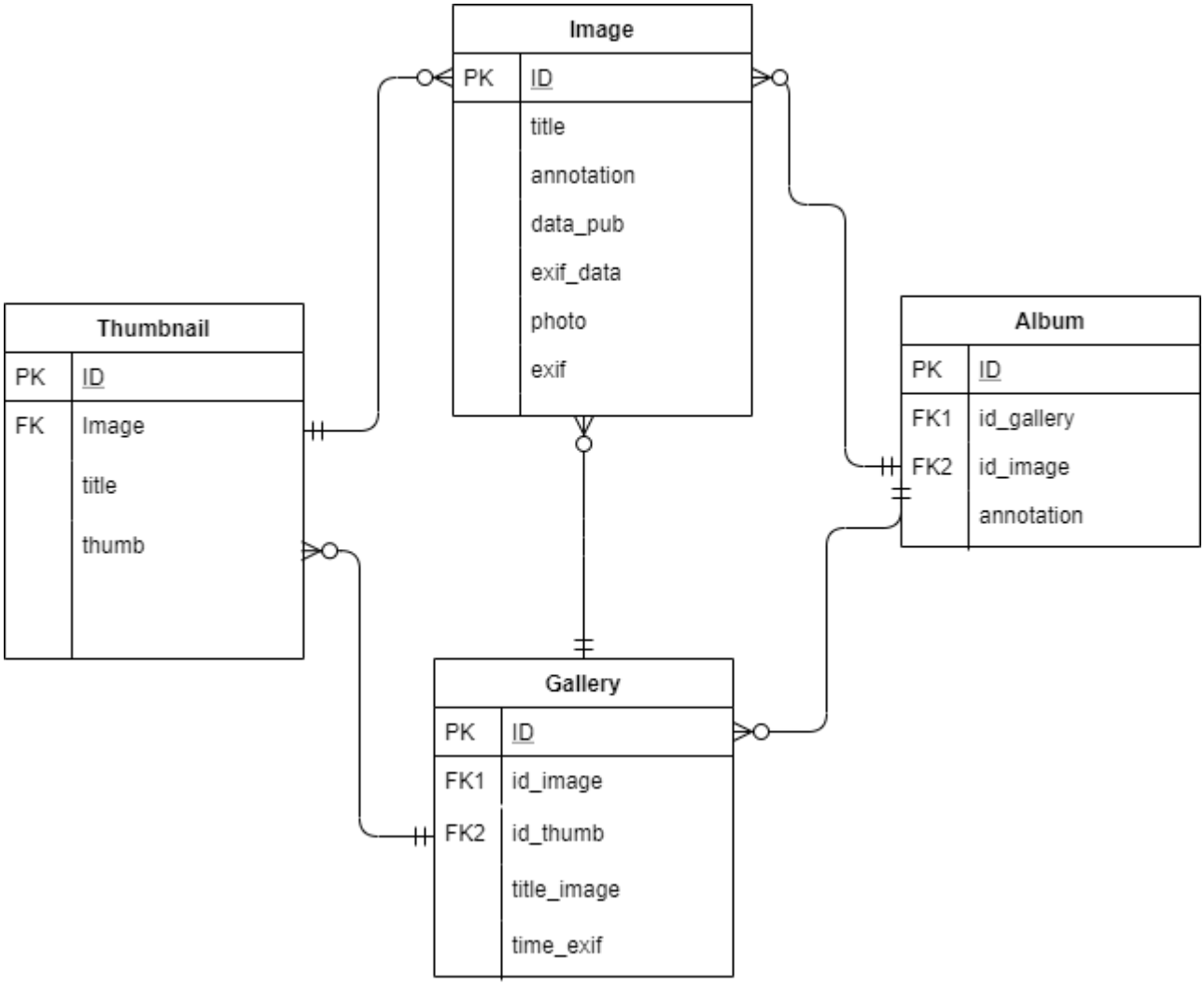

Рисунок 3 – Диаграмма сущностей веб-приложения.

Записи моделей с атрибутами показаны в таблицах 1, 2, 3, 4.

id	title	annotation	data_pub	exif-data	photo	exif
AutoField	CharField	TextField	DateField	TextField	ImageField	ExifField

Таблица 1 – Модель-сущность Image.

id	Image	title	thumb
AutoField	ForeignKey	CharField	ImageField

Таблица 2 – Модель-сущность Thumbnail.

id	id_image	id_thumb	title_image	time_exif
AutoField	ForeignKey	ForeignKey	CharField	ImageField

Таблица 3 – Модель-сущность Gallery.

id	id_gallery	id_image	annotation
AutoField	ForeignKey	ForeignKey	CharField

Таблица 4 – Модель-сущность Album.

2.4 Диаграмма развертывания/компонентов веб-приложения

Разрабатываемое веб-приложение построено на основе архитектуры «клиент-сервер», главной особенностью которой является распределение задач через Интернет (по протоколам TCP/IP) между узлами: серверами, выступающими в роли поставщиков услуг, и клиентами−потребителями услуг. Серверная часть состоит из программного интерфейса приложения− API, для создания которого было решено использовать фреймворк ASP.NET WebAPI; библиотеки функций по работе с базой данных и сервера баз данных под управлением СУБД SQL Server. Клиентская часть представляет собой интерфейс пользователя, отображаемый при помощи веб-браузера.

Диаграмма развёртывания (Deployment diagram) в UML моделирует физическое развертывание артефактов на узлах. Например, чтобы описать веб-сайт, диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают

на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом. [6]

В веб-приложении и клиент и сервер могут быть на одном компьютере и работать локально. Но также есть возможность установки сервера отдельно, так как взаимодействие, вывод м ввод данных, загрузка изображений происходит через внутренние классы Django HttpRequest и HttpResponse методами POST и GET. Базу данных тоже можно заменить. Django поддерживает все популярные СУБД. Диаграмма развертывания веб-приложения показана на рисунке 4.

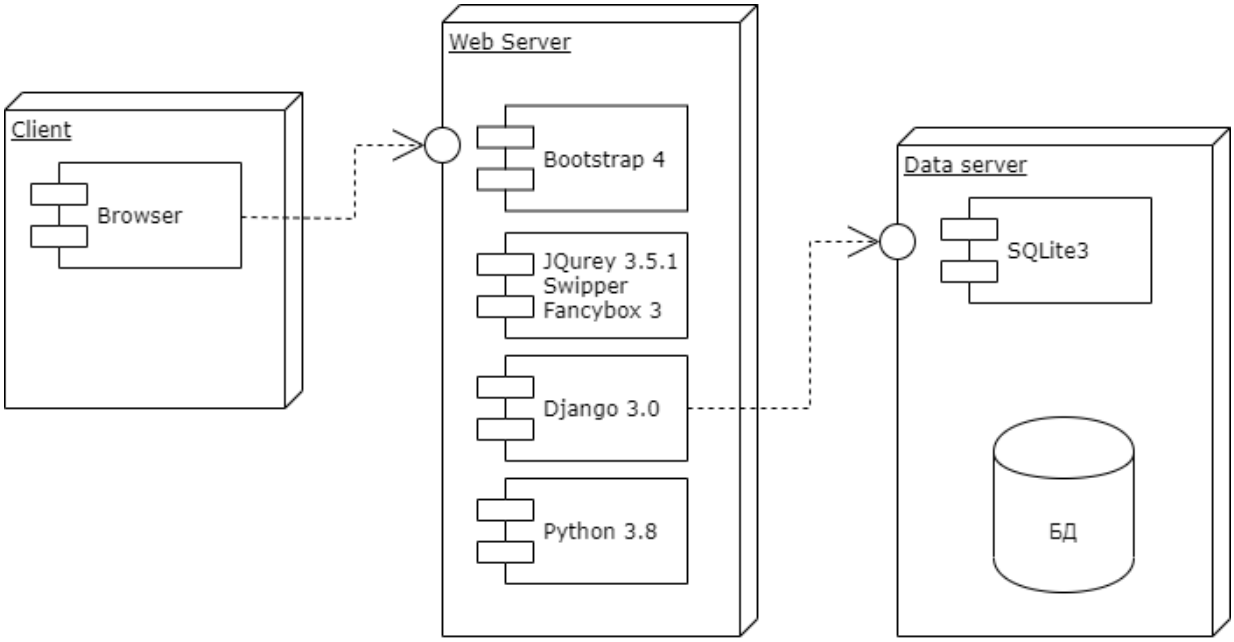

Рисунок 4 – Диаграмма развертывания/компонентов веб-приложения.

2.5 Вывод по главе 2

В главе 2 выполнено проектирование разрабатываемого веб-приложения по созданию фотоальбома–отчета о событии согласно методологии ICONIX.

- Создана диаграмма прецедентов и представлены цели с участием пользователя;
- Сформирован прототип сайта;

- Представлена диаграмма моделей-сущностей и описаны связи между ними;
- Показана диаграмма развертывания веб-приложения.

ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ

Программная реализация системы заключается в создании клиентской части веб-приложения, реализующей пользовательский интерфейс, и серверной части. Все будет выполнятся на фреймворке Django с использованием для вывода дополнительных фреймворков Bootstrap, Jquery с плагинами Swipper и Fancybox.

3.1 Разработка интерфейса пользователя

Пользовательский интерфейс разработан с помощью языка разметки HTML5, каскадных таблиц стилей CSS3 и фреймворка Bootstrap.

HTML — язык гипертекстовой разметки. Аббревиатура образовалась от первых букв английских слов HyperText Markup Languge. HTML применяется для разметки веб-страниц. Она нужна браузерам, которые преобразуют гипертекст и выводят на экран страницу в удобном для человека формате.

CSS — язык описания стилей. Аббревиатура образовалась из первых букв английских слов Cascading Style Sheets — каскадные таблицы стилей. CSS описывает внешний вид HTML-элементов. То есть разработчики с помощью каскадных таблиц стилей определяют, как должен выглядеть тот или иной элемент на странице.

На рисунке 5. можно увидеть, как выглядит файловая структура статических файлов веб-приложения.

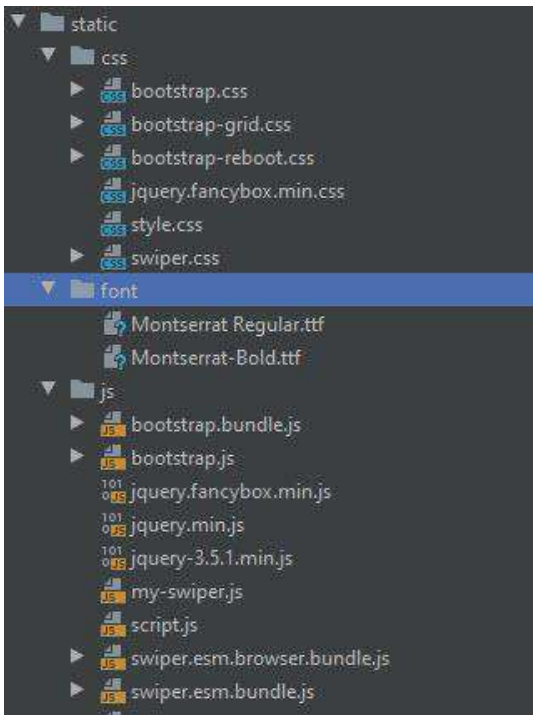

Рисунок 5 – Файловая структура статических файлов веб-приложения.

Папка css содержит файлы стилей, в папке js необходимые скрипты и папка font с используемыми в приложении шрифтами.

Django предоставляет мощный и удобный API для загрузки шаблонов из файловой системы, цель которого — избавиться от избыточного кода в вызовах шаблонов и в самих шаблонах.

Django позволяет динамически генерировать HTML Шаблоны содержат статический HTML и динамические данные, рендеринг которых описан специальным синтаксисом. Шаблоны хранятся в папке Templates. На рисунке 6. можно увидеть, как выглядит файловая структура шаблонов веб-приложения.

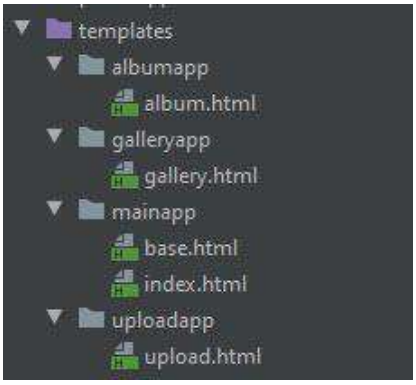

Рисунок 6 – Файловая структура шаблонов веб-приложения.

Base.html содержит в себе базовую структуру html верхней навигационной панелью документа с подключением необходимых css и javascript файлов. На рисунке 7. показано тело base.html документа.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href={% static "css/bootstrap.min.css" %}>
    <link rel="stylesheet" href={% static "css/jquery.fancybox.min.css" %}>
    <link rel="stylesheet" href={% static "css/swiper.css" %}>
    <link rel="stylesheet" href={% static "css/swiper.min.css" %}>
    <link rel="stylesheet" href={% static "css/style.css" %}>
    <title>Графичесческий редактор</title>
</head>
<body>

{% block header %}
{% endblock %}

{% block content %}
{% endblock %}

{% block footer %}
{% endblock %}

<script src={% static "js/jquery-3.5.1.min.js" %}></script>
<script src={% static "js/bootstrap.min.js" %}></script>
<script src={% static "js/swiper.js" %}></script>
<script src={% static "js/swiper.min.js" %}></script>
<script src={% static "js/jquery.fancybox.min.js" %}></script>
<script src={% static "js/my-swiper.js" %}></script>
<script src={% static "js/script.js" %}></script>

</body>
</html>
```

Рисунок 7 – Листинг base.html веб-приложения.

В index.html с помощью тега {% extends 'mainapp/base.html' %} объявляется шаблон, указанный в качестве аргумента как родитель текущего шаблона добавляются блоки для «шапки», «подвала» и основного контента веб-приложения. На рисунке 8. показано тело index.html документа.

```
{% extends 'mainapp/base.html' %}
{% load static %}

{% block header %}
    <main>
        <article>
            <div>
                <a href="/" type="button" class="btn btn-primary btn-lg">Главная</a>
                <a href="/gallery" type="button" class="btn btn-primary btn-lg">Галерея</a>
                <a href="/upload" type="button" class="btn btn-primary btn-lg">Загрузка</a>
            </div>
        </article>
    </main>
    <br>
{% endblock %}

{% block content %}
{% endblock %}

{% block footer %}
    <div class="fixed-bottom">
        <div class="alert alert-primary" role="alert">
            <span id="footer">Footer in construction..  </span>
            <div class="spinner-border" role="status">
                <span class="sr-only">Loading...</span>
            </div>
        </div>
    </div>

{% endblock %}
```

Рисунок 8 – Листинг index.html веб-приложения.

Остальные страницы берут наследование от index.html. Этот способ дает целостность приложению и сайту. Изменяется лишь центральный блок контента для каждой страницы в отдельности. Система шаблонов очень удобна. Это позволяет облегчить работу бэкенд и фронтенд разработчиков. И избавляет от повторения кода.

Все разделы приложения содержат навигационную панель и кнопки главной страницы и страницы галереи. Со стартовой страницы рисунок 9. переходим в раздел «Галерея» рисунок 10.

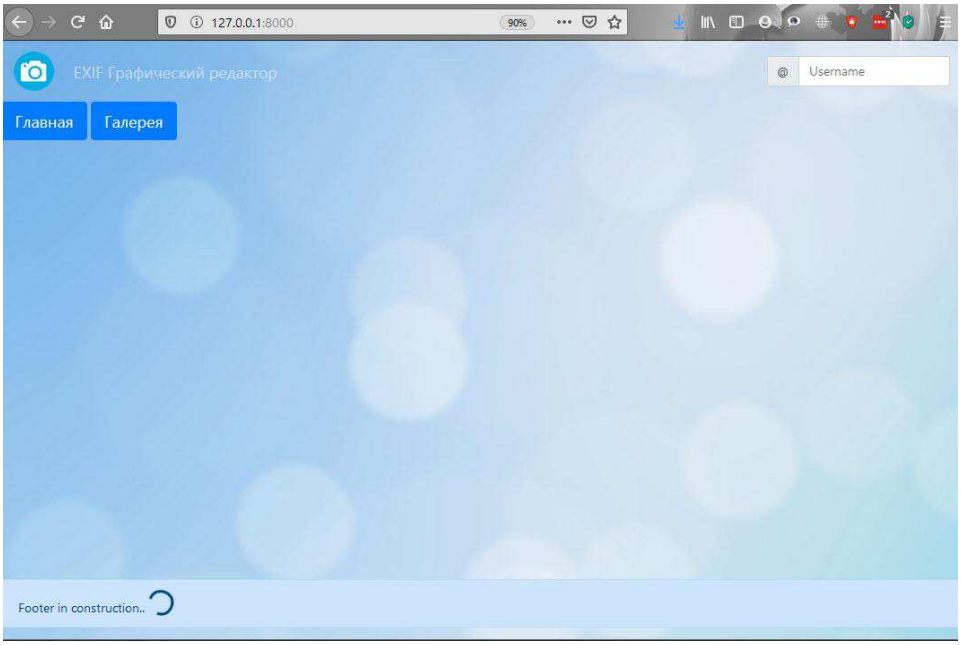

Рисунок 9 – Внешний вид главной(index.html) страницы.

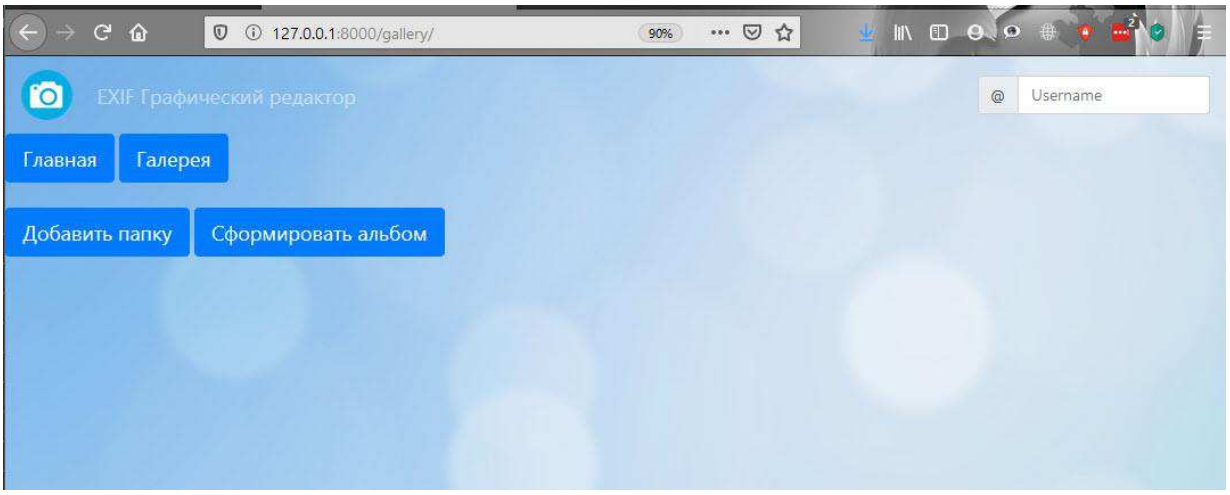

Рисунок 10 – Внешний вид страницы галерея(gallery.html).

После нажатия кнопки «Добавить папку» переходим на страницу загрузки файлов рисунок 11. В Django для этого создали класс форму LocationForm ссылающеюся на класс Image в которой указали нужные на поля для вывода и заполнения. Форму можно увидеть на рисунки 12. Для вывода формы в шаблон используем код на рисунке 13.

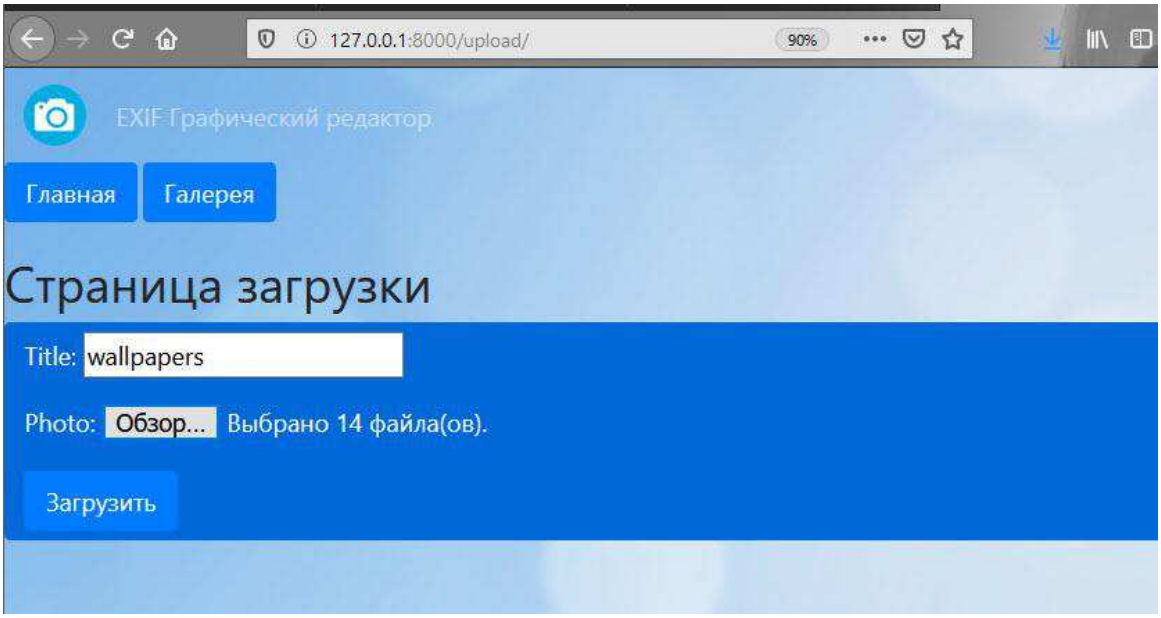

Рисунок 11 – Внешний вид страницы загрузки(upload.html).

```
class LocationForm(forms.Form):
    name = forms.CharField(label=u'Локация')
    photos = forms.ImageField(label=u'Фотографии', widget=forms.FileInput(attrs={'multiple': 'multiple'}))

    class Meta:
        model = Image
        fields = ('name', 'photos')
```

Рисунок 12 – Листинг кода LocationForm.

```
<form method="post" enctype="multipart/form-data" class="btn-primary btn-lg" >
        {% csrf_token %}
       {{ form.as_p }}

        <button type="button" class="btn btn-primary btn-lg" >Загрузить</button>
    </form>

    {% if img_obj %}
        <h3>Succesfully uploaded : {{ img_obj.title }}</h3>
        <img src="{{ img_obj.image.url }}" alt="connect" style="max-height:300px">
    {% endif %}
```

Рисунок 13 – Листинг кода вывода формы загрузки(upload.html).

После чего формируется галерея из загруженных фотографий рисунок 14.

Рисунок 14 – Внешний вид страницы галерея с загруженными фотографиями.

```
<div class="flex2"></div>
    <div>
        <div class="slider-container">
            <div class="swiper-container">
                <div class="swiper-wrapper">
                    {% for img in thumb_gal %}
                        <div class="swiper-slide card">
                            <a data-fancybox="images" href="{{ img.thumb.url }}">
                                <img src="{{ img.thumb.url }}" width="100%" height="100%"/>
                            </a>
                            <br>
                        <input type="checkbox">
                        </div>
                    {% endfor %}
                </div>
                <div class="swiper-pagination"></div>
            </div>
```

Рисунок 15 – Листинг кода вывода галереи с применением fancybox и swipper (gallery.html).

После нажатия кнопки «Сформировать альбом» происходит переход на страницу album.html в которой показываются выбранные фотографии рисунок 16. В которой можно заполнить Аннотацию и нажатием кнопки «Сохранить альбом» выгрузить выбранные фотографии.

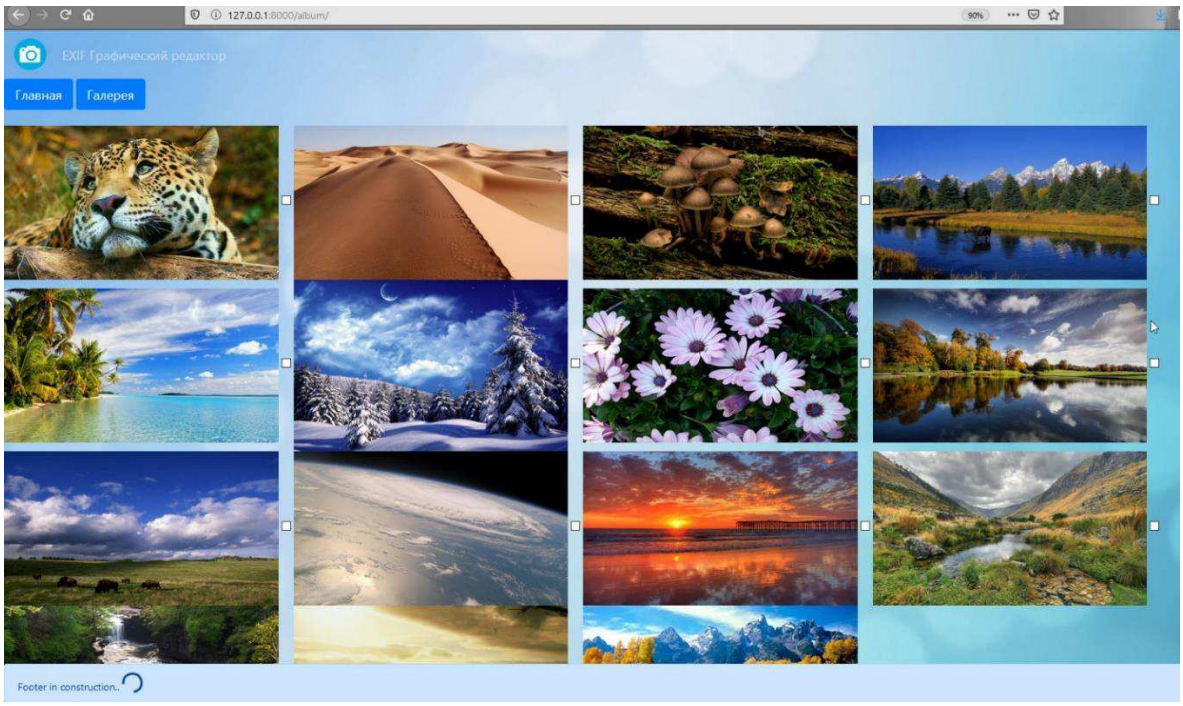

Рисунок 16 – Внешний вид страницы загрузки(album.html).

3.2 Серверная часть

Серверная часть, построена по паттерну MVC(Model-View-Controller). В Django начинается небольшая путаница, поскольку в его терминах этот паттерн называется Model-View-Template рисунок 17.

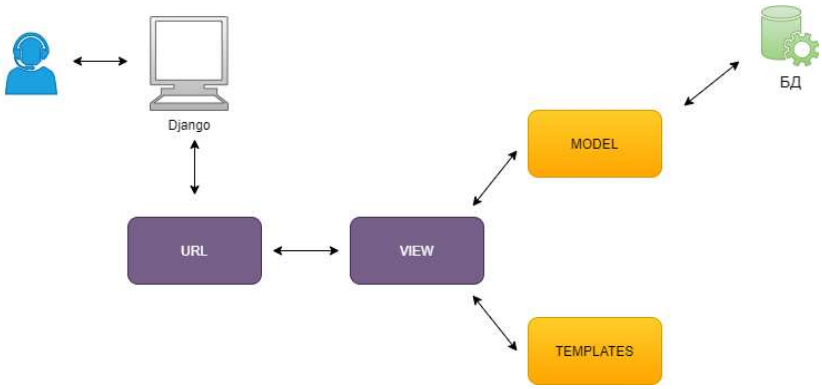

Рисунок 17 – Представление Model-View-Template.

URL-шаблоны позволяют связывать URL с представлениями. Шаблон URL состоит из шаблона строки, представления и имени (опционально), с

помощью которого можно задать имя для URL всего проекта. Django перебирает каждый шаблон и останавливается на первом, который соответствует запрошенному URL. Затем библиотека импортирует представление совпавшего URL-шаблона и исполняет его, передавая экземпляр класса HttpRequest и ключевое слово или позиционные аргументы. [7]

На рисунке 18. Представлен urls.py корневой папки веб-приложения. После перехода по одному из пути, дальше размаривается urls.py внутри отдельных приложений.

```
from django.contrib import admin
from django.urls import path, include
from django.conf.urls.static import static

from photoapp import settings

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('mainapp.urls')),
    path('upload/', include('uploadapp.urls')),
    path('gallery/', include('galleryapp.urls')),
    path('album/', include('albumapp.urls'))
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
                          document_root=settings.MEDIA_ROOT)
```

Рисунок 18 – Листинг кода urls.py корневой папки приложения.

Views отвечают за обработку и передачу данных и могут реализовывать одну или несколько функций. В Django представления views являются своеобразными доставщиками контента от модели к шаблону отображения. Реализуются простыми функциями и методами языка Python. На рисунке 19. можно увидеть функцию загрузки фотографии.

```
def create_to_img(request):
    user = request.user
    if request.method == 'POST':
        form = SaveForm(request.POST)
        file_form = SaveImgForm(request.POST, request.FILES)
        files = request.FILES.getlist('file')
        if form.is_valid() and file_form.is_valid():
            feed_instance = form.save(commit=False)
            feed_instance.user = user
            feed_instance.save()
            for f in files:
                file_instance = SaveImg(file=f, feed=feed_instance)
                img_obj = file_instance.save
                return render(request, 'uploadapp/upload.html', {'form': form, 'img_obj': img_obj})
    else:
        form = SaveForm()
        file_form = SaveImgForm()
        return render(request, 'uploadapp/upload.html', {'form': form})
```

Рисунок 19 – Листинг кода функции загрузки фотографии.

Модель является единственным источником информации о данных. Она содержит основные поля и поведение данных, которые вы храните. Как правило, каждая модель отображается в одну таблицу базы данных. Django следует паттерну «Active Record».

Active Record — Активная запись это шаблон проектирования. Упрощает создание и использования бизнес объектов, данные которых требуют хранения в базе данных. В MVC это буква M – Модель.

После создания модели рисунок 20. производиться миграция в базу данных рисунок 21, при миграции Django сам добавляет первичный ключ (primary key).

Первичный ключ — это ограничение позволяющее однозначно идентифицировать каждую запись в таблице SQL. По первичному ключу остальные модели обращаются к записи, с помощь создание связей один-к-одному, используется внешний ключ (foreign key). Так же есть отношения, многие-ко-многим (Many to Many) и один-ко-многим (One to Many).

```python
class Image(models.Model):
    title = models.CharField(max_length=50)
    description = models.CharField(max_length=150, blank=True, null=True)
    annotation = models.TextField(blank= True, null=True )
    data_pub = models.DateTimeField(auto_now=True)
    photo = models.ImageField(upload_to='photos')
    exif_data = models.DateTimeField(auto_now=True, null=True)

    def __str__(self):
        return self.title


class Thumbnail(models.Model):
    key =models.ForeignKey(Image, on_delete = models.CASCADE)
    title = models.CharField(max_length=20, null=True)
    thumb = models.ImageField(upload_to='thumbnail')
    date_pub = models.DateTimeField()

    def __str__(self):
        return self.title
```

Рисунок 20 –Класс Thumbnail ссылается по внешнему ключу на класс Image.

```python
class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Image',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('title', models.CharField(max_length=50)),
                ('description', models.CharField(max_length=150, null=True)),
                ('annotation', models.TextField(null=True)),
                ('data_pub', models.DateTimeField()),
                ('photo', models.ImageField(upload_to='photos')),
            ],
        ),
        migrations.CreateModel(
            name='Thumbnail',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('title', models.CharField(max_length=20, null=True)),
                ('thumb', models.ImageField(upload_to='thumbnail')),
                ('date_pub', models.DateTimeField()),
            ],
        ),
    ]
```

Рисунок 21 –Миграция классов в Базу данных.

В веб-приложении работа происходит с EXIF данными фотографии. EXIF (Exchangeable Image File Format) – это часть файла, где хранятся метаданные фотографии, такие как дата создания, тип устройства, скорость, диафрагма и многие другие.

EXIF данные представляются в json формате. Для приравнивания нужных нам атрибутов в классе, мы «денормализуем» данные о Модели, устройства которым была сделана фотография и Дату создания фотографии. На рисунке 20. в класс Image добавляем атрибут exif и с помощью библиотеки Exiffield «денормализуем» json данные и методом exifgetter считываем записи по ключам Model и DateTimeOriginal.

```
class Image(models.Model):
    title = models.CharField(max_length=40)
    annotation = models.TextField(blank=True, default='')
    data_pub = models.DateTimeField(auto_now=True, null=True)
    photo = models.ImageField(upload_to='photos')
    camera = models.CharField(editable=False, max_length=100, blank=True)
    data_exif = models.TextField(editable=False, max_length=50, blank=True)

    exif = ExifField(source='photo',
                     denormalized_fields={'camera': exifgetter('Model'),
                                          'data_exif': exifgetter('DateTimeOriginal')
                                          },
                     )

    def __str__(self):
        return self.title
```

Рисунок 20 - EXIF данные DateTimeOriginal в виде списка.

3.3 Вывод по главе 3

В третьей глава описан процесс разработки веб–приложение по созданию фотоальбома–отчета о событии. Рассмотрены клиентская и серверная части системы, их компоненты и взаимосвязи между ними. Показаны Классы для предоставления сущностей из базы данных. На примерах показаны работы методов.

Как результат, в соответствии с моделью системы полученной в главе 2, реализовано веб-приложение для создания фотоальбома–отчета о событии.

Для демонстрации результатов работы на иллюстрациях представлен графический интерфейс пользователя, описан процесс оформления предзаказа.

ЗАКЛЮЧЕНИЕ

В ходе выполнения ВКР были сформированы требования к системе, изучена информация о данной предметной области и проанализированы средства для реализации однопользовательского автономного веб-приложения реализующее интерактивный инструмент подготовки фотоальбома–отчет о событии. В результате формирований требований к системе была разработана общая архитектура системы и выбрана клиент-серверная модель организации системы.

Согласно ICONIX методологии были проведены анализ системы, сформированы Диаграммы прецедентов, сущностей и каркас пользовательского интерфейса.

В соответствии с функциональными требованиями заказчика и выводами по результатам обзорного исследования сервисов-аналогов, разработан проект веб-приложения, реализующего веб-интерфейс сервиса для создания фотоальбома–отчета о событии. В качестве результат проектирования приведена модель системы в виде совокупности диаграмм на языке моделирования UML, а также прототипа графического интерфейса веб-приложения. Описана архитектура приложения, компонентная структура, функциональный возможности. Выделены и представлены сущности из базы данных необходимые для работы приложения, описаны внесенные в базу изменения.

Как результат, выполнена программная реализация веб-приложения по созданию фотоальбома–отчета о событии.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SQLite, MySQL и PostgreSQL: сравниваем популярные реляционные СУБД [Электронный курс] – Режим доступа: https://tproger.ru/translations/sqlite–mysql–postgresql–comparison/
2. PyCharm [Электронный курс] – Режим доступа: https://ru.wikipedia.org/wiki/PyCharm
3. Bootstrap что это и как его установить [Электронный курс] – Режим доступа: https://codelessons.ru/bootstrap–3/bootstrap–dlya–novichkov–chto–eto–i–kak–ego–ustanovit.html
4. ICONIX [Электронный курс] – Режим доступа: https://studbooks.net/2138987/informatika/iconix
5. Диаграммы использования [Электронный курс] –Режим доступа: https://www.sites.google.com/site/anisimovkhv/learning/pris/lecture/tema12/tema12_2
6. Диаграмма развёртывания [Электронный курс] – Режим доступа: https:////ru.wikipedia.org/wiki/Диаграмма_развёртывания
7. Добавление URL-шаблонов в представления [Электронный курс] – Режим доступа: https://pythonru.com/primery/blog-na-django-15-dobavlenie-url-shablonov-v-predstavlenija
8. СТО 4.2-07-2014 Стандарт организации «Общие требования к построению, изложению и оформлению документов учебной деятельности. – Красноярск : ИПК СФУ, 2014. – 60 с.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия
« _____» _____ 2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – «Информатика и вычислительная техника»
код – наименование направления

Веб–приложение по созданию фотоальбома–отчета о событии
тема

Руководитель	подпись, дата	канд. техн. наук, доцент должность, ученая степень	А. И. Постников инициалы, фамилия
Консультант	26.06.2020 подпись, дата	ст. преподаватель должность, ученая степень	О. В. Шмелев инициалы, фамилия
Выпускник	26.06.2020 подпись, дата		Е. В. Абросимов инициалы, фамилия
Нормоконтролер	подпись, дата	канд. техн. наук, доцент должность, ученая степень	А. И. Постников инициалы, фамилия

Красноярск 2020