МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Кафедра информационных технологий управления предприятием

Курсовая работа по курсу  
«Технологии программирования»

Web-приложение «Складской учёт коммерческой организации»

Выполнили: студенты 3 курса, группы 3.2

Кицуль Д.Ю., Мельникова И.Н.,Тоншин Д.В.

Руководитель: Полещук Х. А.

Воронеж 2019

1. Введение

Склады являются важными звеньями технологического процесса промышленных предприятий, а для оптовой и розничной торговли они служат фундаментом, поэтому склады предприятий, намеренных опережать конкурентов, требуют современной организации, современных технологий и квалифицированных кадров.

Долгое время недооцененной оставалась существенная функция складов по накоплению и формированию нужного потребителям ассортимента Товаров. Предприятия часто видят свою главную цель в налаживании эффективных и экономичных процессов снабжения, производства и потребления, не придавая должного значения внутренним складским операциям.

Вопросы организации деятельности товарного и складского хозяйств на предприятии являются достаточно важными в условиях рыночной экономики. Движение материальных потоков в логистической цепи невозможно без концентрации в определенных местах необходимых запасов, для хранения которых предназначены соответствующие склады. Складирование играет важнейшую роль в управлении цепочками поставок.

Особый акцент необходимо делать на планировании повседневной работы склада: приемке, хранении, отборке, упаковке, комплектации и распределении заказов покупателей и потребителей.

В процессе товародвижения невозможно переоценить роль складов, ведь без них нельзя представить себе основное решение задачи логистики, которая состоит в том, чтобы процесс передвижения товаров от производителя к потребителю был максимально эффективным при минимальных затратах на осуществление логистических операций. Именно этим обусловлена актуальность темы курсовой работы.

1. Анализ предметной области
   1. Глоссарий

Для конкретного понимания темы работы составлен небольшой справочник узкоспециализированных слов. Ниже представлены понятия и определения к ним.

1. **Склад — это** здания, сооружения и разнообразные устройства, предназначенные для приемки, размещения и хранения поступивших товаров, где выполняются работы по приемке, подсортировке, хранению, фасовке, отпуску товаров.
2. **Работник склада** – сотрудник склада, ведущий учет материальных ценностей.
3. **Контрагент** - это физическое или юридическое лицо, выступающее одной из сторон сделки. Под контрагентами понимаются различные лица, предприятия и учреждения, с которыми организация вступает в торговые, финансовые, гражданско-правовые и другого рода отношения.
4. **Продавец** – лицо (физическое или юридическое), которое продает товар или оказывает услугу.
5. **Покупатель** – лицо (физическое или юридическое), которое является приобретателем (осуществляет оплату деньгами) товара или услуги
6. **Накладная** - документ первичного бухгалтерского учета предприятия, с помощью которого производится отгрузка и доставка продукции от компании-продавца покупателю, и на основании которого разрешается списание проданного товара с баланса предприятия
7. **Реализация** товаров со склада в базе оформляется документом **Расходная накладная**.
8. **Приходная накладная** применяется для учета поступления материальных ценностей внутри организации. Она хранится на складе и используется для складского учета до момента её проведения.
   1. Методология

В разработке программного обеспечения мы используем методику Agile. Agile является современной методикой для создания качественного, соответствующего требованиям заказчика программного обеспечения, и за меньшее время. Agile делает проект менее громоздким и более прозрачным.

В первую очередь стоит отметить, что Agile - управление очень гибкое. Если, например, традиционная методология указывает на конкретные этапы работы, то Agile легко подстраивается под потребителя конечного продукта и требования заказчика.

В конечном продукте число дефектов минимизируется, ведь он является результатом тщательной проверки качества, которая проводится по завершении каждого этапа-спринта.

Кроме того, Agile быстро запускается, легко реагирует на изменения, позволяет команде разработчиков и клиентов поддерживать постоянную связь в реальном времени.

* 1. Аналоги решения задачи

Приложение **1С 8.3 Бухгалтерия** содержит модуль управления складами, который позволяет решать задачу такого типа в коммерческой деятельности организаций. Это универсальная программа массового назначения для автоматизации бухгалтерского и налогового учета, включая подготовку обязательной (регламентированной) отчетности. Это готовое решение для ведения учета в организациях, осуществляющих любые виды коммерческой деятельности: оптовую и розничную торговлю, комиссионную торговлю (в том числе субкомиссию), оказание услуг, производство и т.д. Кроме того, с помощью "1С:Бухгалтерии 8" могут вести учет индивидуальные предприниматели, применяющие упрощенную систему налогообложения или общий режим налогообложения.

## ÐÐ¾ ÑÐ¼Ð¾Ð»ÑÐ°Ð½Ð¸Ñ ÑÑÑÐ°Ð½Ð¾Ð²Ð»ÐµÐ½ ÐÑÐ½Ð¾Ð²Ð½Ð¾Ð¹ ÑÐºÐ»Ð°Ð´

Рисунок 1 – Диаграмма вариантов использования

Минус этой системы заключается в усложнённой модели учета товаров с лишними действиями и бюрократией. Система достаточно непроста для овладения её начинающим пользователям и подразумевает наличие некоторых знаний компьютерной бухгалтерии, тогда как пользователю может быть это вовсе не нужно.

* 1. Анализ моделируемой системы

### 2.4.1. Диаграмма вариантов использования

отражает отношения между актерами и прецедентами и является исходным концептуальным представлением системы в процессе ее проектирования и разработки.

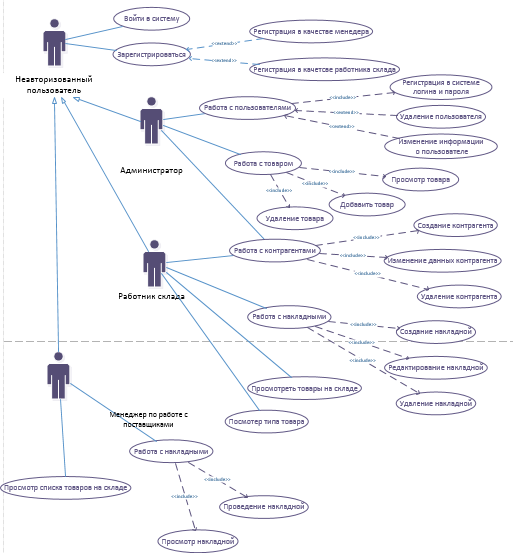


Рисунок 2 – Диаграмма вариантов использования

Рассмотрим каждый вариант использования более подробно:

**Пользователи системы:** работник склада, администратор, менеджер по работе с контрагентами.

Работник склада решает следующие задачи:

- прием продукции по приходной накладной.

- реализация продукции по расходной накладной.

Администратор решает следующие задачи:

* добавление пользователя с присвоением уникального логина и пароля;
* изменение данных о пользователе;
* удаление пользователя из списка.
* добавление, изменение, удаление товара из базы (номенклатуры).
* добавление, изменение, удаление контрагента из базы.

Менеджер по работе с контрагентами решает следующие задачи:

* добавление, изменение и удаление поставщиков из базы данных;
* добавление и удаление накладных из базы данных;
* просмотр актуального списка поставщиков.
* просмотр номенклатуры с включенными в неё товарами
* просмотр актуального списка товаров на складе

### 2.4.2 Анализ процесса входа в систему

На рисунке 3 представлена диаграмма последовательности, иллюстрирующая вариант использования «Войти в систему». При запуске приложения пользователем ему автоматически выводится форма входа, которая запрашивает ввод логина и пароля для работы с соответствующим окном приложения. Логин и пароль выдается либо администратором, либо создается пользователем при регистрации. В случае правильного ввода логина и пароля открывается соответствующая роли пользователя страница приложения. В случае неправильного ввода система выводит страницу с сообщением об ошибке.



Рисунок 3 – Вариант использования «Вход в систему»

На рисунке 4 представлен вариант использования «Получение списка накладных». При нажатии на кнопку «Накладные» посылается запрос на сервер о том, что нужно обратиться в базу данных. Сервер получает затем результат запроса из БД и возвращает его приложению. Приложение в предусмотренном для пользовательского интерфейса виде отображает список накладных.



Рисунок 4 – Вариант использования «Получение списка накладных»

### 2.4.3 Диаграмма классов

На рисунке 5 представлена диаграмма классов проектируемого приложения. Она коррелирует с сущностями, так как функциональная составляющая приложения предполагает тесное взаимодействие с базой данных.

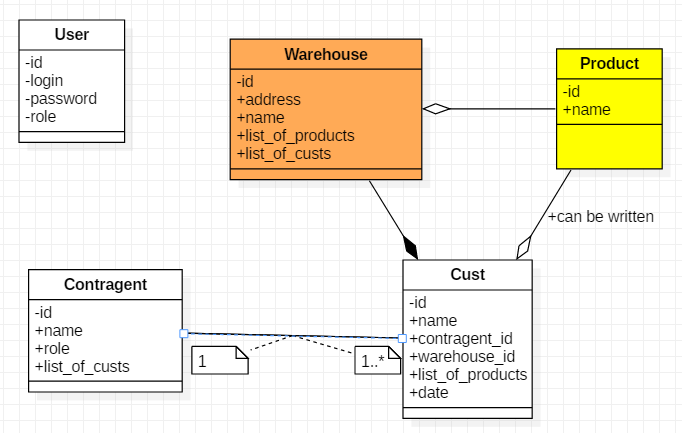
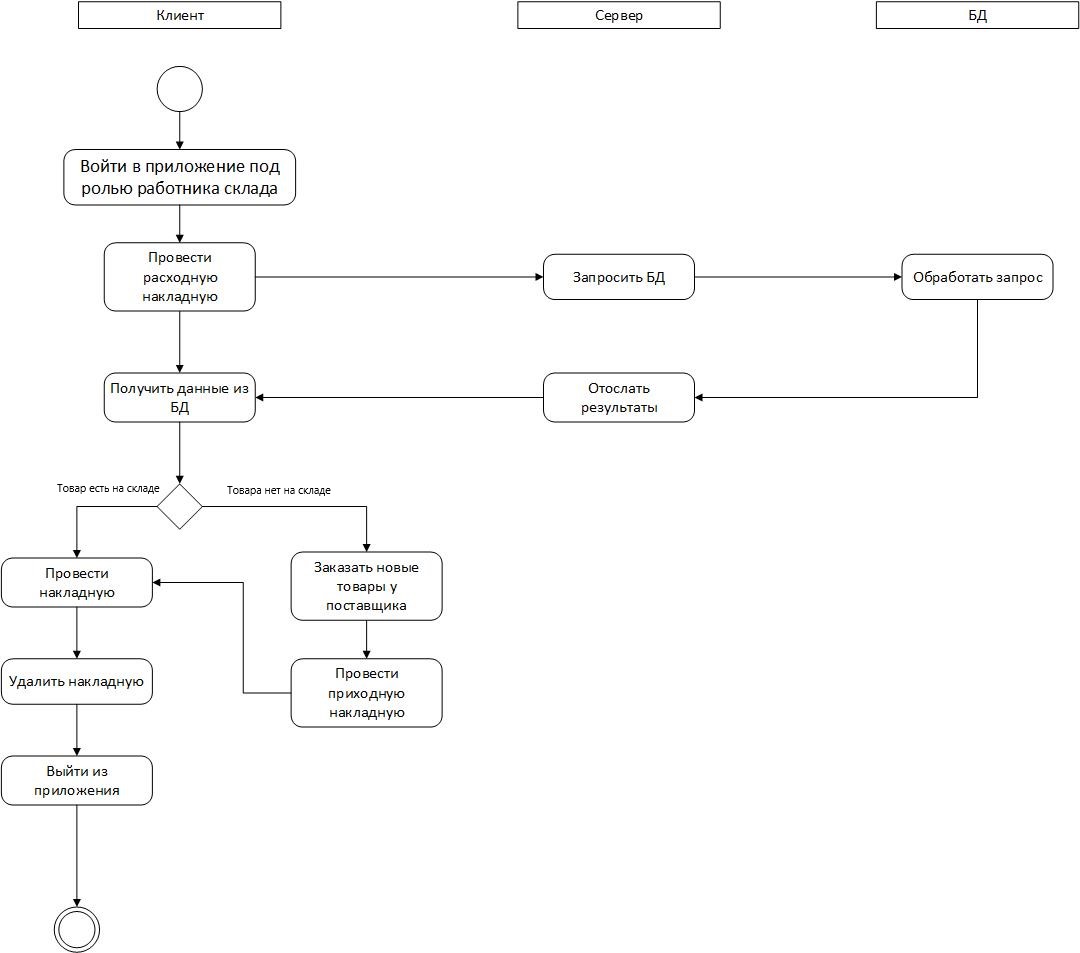


Рисунок 5 – Диаграмма классов приложения

В данной диаграмме классы представляют собой объектные модели, описанные в файле программы classes.py. Эта модель (Object Relationship Model) позволяет на основе соответствия по имени таблиц (сущностей) и полям из базы данных делать мэппинг на них классов, которыми в дальнейшем при успешном построении модели можно будет управлять (совершать запросы и т.д.) с помощью средств используемой в программе библиотеки для работы с БД SQLAlchemy. Стоит заметить, что для отношения многие ко многим, где, как известно, создаётся третья таблица с составным ключом из id двух взаимодействующих таблиц, не предусмотрено создание классов. Поэтому данная диаграмма не отражает таблицы product\_in\_cust и product\_on\_wh, которые соответствующие продуктам в накладной и продуктам на складе.

### 2.4.4 Диаграмма деятельности

На рисунке 6 представлена диаграмма деятельности, демонстрирующая процесс проведения расходной накладной (кроме работника склада). Для отдельных ролей некоторые функции могут быть недоступны. Важным является тот факт, что расходная накладная может формироваться лишь из товаров на складе и ниоткуда более. После проведения накладная удаляется и товаров на складе становится меньше ровно на то количество, которое было прописано в расходной накладной.



### 2.4.5 Диаграмма коммуникаций

Диаграмма коммуникаций демонстрирует последовательность взаимодействия пользователя, приложения, включающего в себя клиент и сервер, а также базы данных, в которую подают запросы. В данном случае представлен обобщенный процесс добавления пользователем объекта в базу данных.

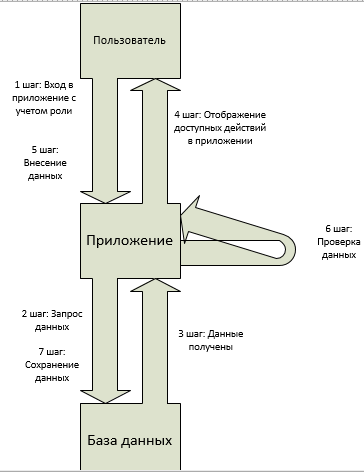


Рисунок 7 – Диаграмма коммуникаций «Добавление объекта»

На рисунке 8 представлен тот же тип диаграммы с подробным описанием взаимодействия модулей системы для процесса изменения данных о контрагенте в приложении.



Рисунок 8 – Диаграмма коммуникаций «Изменение контрагента»

### 2.4.5 Диаграмма состояний

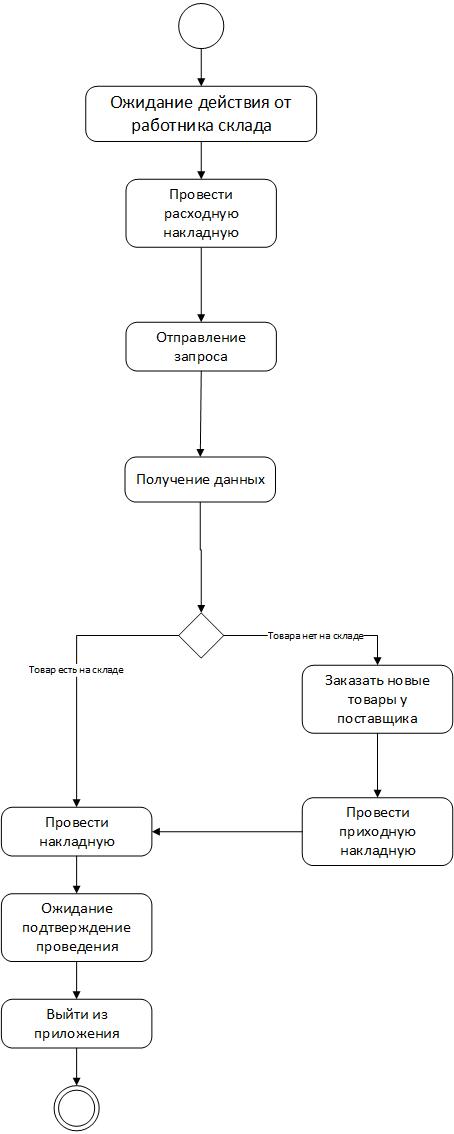


Рисунок 9 – Диаграмма состояний

Выше на рисунке 9 представлена диаграмма состояний приложения для рассмотреного процесса проведения расходной накладной.

### 2.4.5 Диаграмма объектов

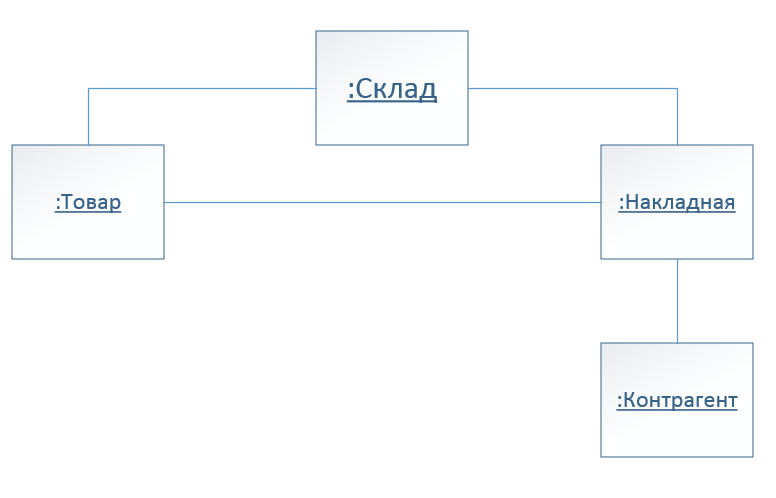


Рисунок 8 – Диаграмма объектов

На рисунке 8 представлены объекты приложения, взаимодействующие друг с другом

### 2.4.7 Диаграмма развертывания.

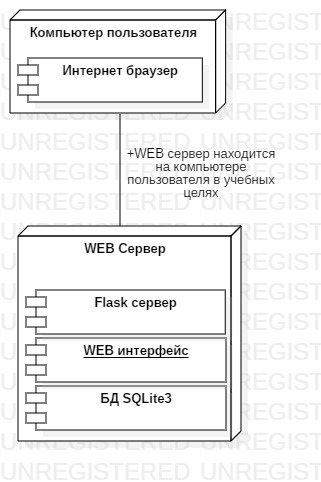


Рисунок 10 – Диаграмма объектов

Диаграмма развертывания показывает узлы выполнения программных компонентов реального времени, а также процессов и объектов, которые будут рассмотрены подробнее в следующей главе.

1. Постановка задачи

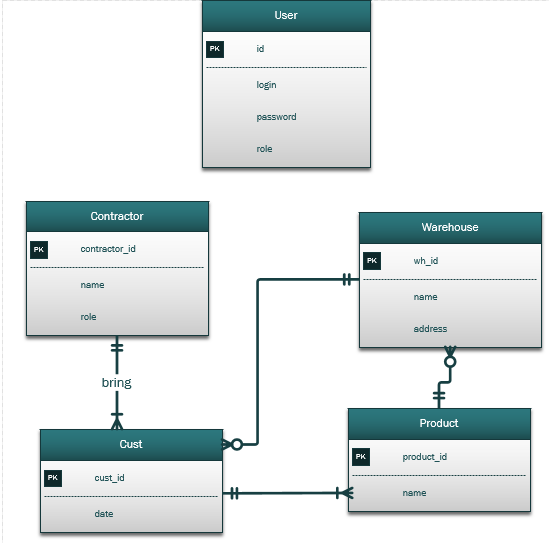
Целью данной работы является разработка web-приложения, позволяющего вести простой учет приходящих на склад и отгружаемых со склада товаров, оптимизация работы складов предприятия, контроля всех приходящих и уходящих товаров, а также уменьшение временных ресурсов во всех возможных операциях предприятия.

Для этого программа должна обладать следующими функциями:

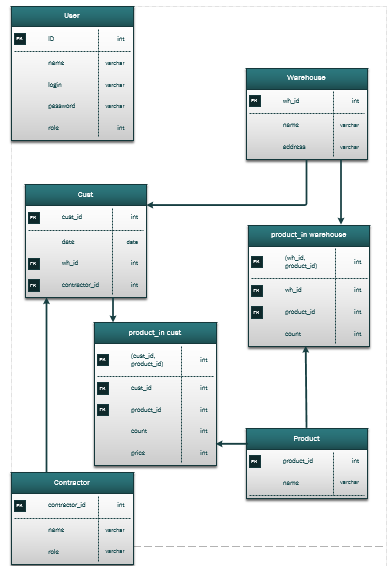
1. Осуществление авторизации пользователя;
2. Осуществление регистрации нового пользователя либо добавление пользователя администратором.
3. Наличие нескольких типов пользователей (администратор, работник склада, менеджер по работе с контрагентами);
4. Ведение учета о приходе или расходе товаров;
5. Возможность внесения изменений определёнными видами пользователей;
6. Хранение всей необходимой информации о покупателях, поставщиках, накладных и наличии товаров на складе в базе данных;
7. Наличие интуитивно понятного для пользователя интерфейса;
8. Проектирование

4.1 База данных приложения

ER-диаграмма взаимодействия сущностей базы данных для приложения.



Физическая модель базы данных



1. Таблица Warehouse – основная сущность работы приложения. Имеет атрибуты name - название и address - адрес.
2. Таблица User – таблица для хранения пользователей в базе данных. При регистрации или при добавлении администратором в базе создаётся автоматически запись о новом пользователе с полями login – логин, password – пароль и роль в системе - role.
3. Таблица Cust – является таблицей, отображающей сущность «Накладная». Таблица содержит атрибуты, ссылающиеся на первичные ключи склада и контрагента, что соответствует связи один ко многим.
4. Таблица Contractor - отображает сущность «Контрагент». Содержит атрибут name – имя контрагента.
5. Таблица Product - отображает сущность «Товар». Аналогично с предыдущей таблицей содержит одно поле name – название товара.
6. Производная таблица product\_in\_cust отображает наличие конкретного товара в накладной. Атрибут price устанавливает цену товара в накладной, count – его количество. Стоит отметить, что при удалении накладной все записи со ссылкой на накладную в таблице product\_in\_cust удаляются каскадно.
7. Производная таблица product\_on\_wh отображает наличие конкретного товара на складе. Атрибут count устанавливает количество товара на складе.

4.2 Средства реализации

Для реализации данного проекта был выбран язык Python. Он удобен в использовании как обновляемый, поддерживаемый язык с большим количеством библиотек.

Это очень важно в данном проекте, поскольку в нем должны быть реализованы различные модули: взаимодействия с базой данных, модуль запуска сервера, модули взаимодействия с динамическим содержимым в HTML шаблонах и так далее. Python позволяет максимально упростить взаимодействие между back-end и front-end сторонами проекта. Это то, что необходимо в учебном проекте.

В качестве основы приложения, его каркаса был выбран Python веб-Фреймворк Flask. Он очень гибкий в работе. Это постоянно обновляемый фреймворк, поэтому ошибки в его реализации быстро исправляются.

Flask в свою очередь использует инструменты Werkzeug и Jinja2. Последний представляет из себя удобный шаблонизатор. Так как в данном проекте есть множество страниц с динамическим содержимым (вывод таблицы со списком тренировок, к примеру), использование кода Jinja2 в HTML шаблонах было очень удачным решением.

Flask - масштабируемый фреймворк, не имеющий жесткой архитектуры. Работа с этим фреймворком позволяет выбирать модули под конкретные задачи и устанавливать их по мере необходимости, без какой-либо жесткой привязки к определенной архитектуре приложения.

Еще одной причиной по которой был выбран язык программирования Python, является то, что под разработку на этом языке было написано множество библиотек по взаимодействию с базами данных.

В данном проекте был выбрана встроенная база данных SQLite3. Основная причина такого выбора – автономность, отсутствие необходимости запускать отдельный сервер для работы с данными из БД. Также SQLIte3 поддерживается большим количеством библиотек по работе с базами данных на Python.

Для взаимодействия с базой данных, для получения из нее данных, их обновления, удаления была использована ORM библиотека SQLAlchemy.

Её преимущества:

* Безопасность. Параметры запросов экранируются, что делает атаки типа внедрение SQL-кода маловероятными.
* Производительность. Повышается вероятность повторного использования запроса к серверу базы данных, что может позволить ему в некоторых случаях применить повторно план выполнения запроса.
* Переносимость. SQLAlchemy, при должном подходе, позволяет писать код на Python, совместимый с несколькими back-end СУБД. Несмотря на стандартизацию языка SQL, между базами данных имеются различия в его реализации, абстрагироваться от которых и помогает SQLAlchemy.

Кроме того, в front-end разработке применены широко используемые и универсальные HTML, CSS, не теряющие своей актуальности с течением времени.

Использование Python позволило с легкостью связать основные модули приложения, связать back-end (взаимодействие с базой данных, логика приложения) и front-end (HTML шаблоны, динамические структуры в HTML) части проекта.

4.3 Функциональная модель

В традиционной модели клиент-серверного взаимодействия, которой мы до сих пор следовали, у нас есть клиент (веб-браузер клиента) посылающий запросы серверу. Запрос просто запрашивает веб страницу, как в случае, когда вы переходите по ссылку «Мой Профиль», или же он может выполнить какое-то действие на сервере, как в случае когда пользователь редактирует свой профиль и нажимает «Отправить». Независимо от типа, сервер отвечает на запрос отправкой новой страницы клиенту прямо или посредством редиректа. Затем браузер заменяет текущую страницу новой. Этот цикл повторяется до тех пор, пока пользователь остается на сайте. Мы называем эту модель «серверной» т. к. сервер выполняет всю работу по генерации страниц, клиент же просто отображает страницы по мере их получения.

