Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Разработка базы данных и веб-приложения для аналитики курса акций с помощью прогнозов различных специалистов.»

Студент _____ИУ7-63Б_____ _____ _____К.Р.Ахметов_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ ____Ю.М.Гаврилова_____
(Подпись, дата) (И.О.Фамилия)

Москва, 2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ___ИУ-7____
(Индекс)
_____ И.В.Рудаков_
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине _____Базы данных_____

Студент группы ___ИУ7-63Б_____

_____Ахметов Карим Ринатович_____
(Фамилия, имя, отчество)

Тема курсового проекта _Разработка базы данных и веб-приложения для аналитики курса акций с помощью прогнозов различных специалистов.

Направленность КП (учебный, исследовательский, практический, производственный, др.)
_____учебный_____
Источник тематики (кафедра, предприятие, НИР) ____кафедра_____

График выполнения проекта: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Задание__Разработать базу данных и веб-приложение, позволяющее пользователю ознакомиться с прогнозами курса акций различных специалистов, хранящимися в базе данных. Реализовать три вида ролей – пользователь, специалист и администратор. Сделать возможным для специалиста создание и изменение прогнозов в базе данных, а для администратора назначение ролей пользователей и абсолютный доступ к базе данных.

Оформление курсового проекта:

Расчетно-пояснительная записка на _20-25_ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
На защиту проекта должна быть представлена презентация. состоящая из 10-15 слайдов ___
На слайдах должны быть отображены: постановка задачи, использованные методы и_____
алгоритмы, расчетные соотношения, структура комплекса программ, интерфейс.

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсового проекта _____ ____Ю. М. Гаврилова
(Подпись, дата) (И.О.Фамилия)
Студент _____ _____К.Р. Ахметов
(Подпись, дата) (И.О.Фамилия)

Оглавление

Введение

В современном мире рынок ценных бумаг играет важную роль в жизни каждого человека. В особенности это касается людей, занимающихся инвестированием или трейдингом. Как правило, у каждого игрока на фондовой бирже есть определенная тактика, которая формируется в зависимости от личных предпочтений и инструментов.

В большинстве случаев фундамент тактики рядового пользователя зиждется на совокупности авторитетных мнений различных специалистов, которые порой могут кардинально различаться. Как следствие, для лучшего понимания мыслимых сценариев изменения курса акций возникает необходимость ознакомления с наибольшим количеством отличных точек зрения.

Завершающим этапом выстраивания плана действий будет выбор наиболее правдоподобных прогнозов по определенным критериям. На практике зачастую важнейшем критерием отбора является авторитетность специалиста для пользователя, которая формируется лишь со временем на основе личного уровня доверия источнику.

На основание вышеизложенного можно выделить ряд проблем, с которыми приходится сталкиваться обычному пользователю:

1) Для получения необходимой информации пользователь вынужден посещать обширное количество интернет-ресурсов;
2) При отсутствии должного опыта пользователю крайне трудно выбрать подходящий прогноз;

Решением первой проблемы выступает агрегация множества прогнозов в единое веб-приложение, которое предоставит информацию конечному пользователю в удобном, доступном виде. При чем важно отметить, что подход агрегации данных частично решает вторую проблему. Имея в распоряжение

огромную базу данных, можно реализовать различные алгоритмы, которые будут в состояние рекомендовать наилучший прогноз для неопытного пользователя. Какие именно алгоритмы будут лежать в основе предлагаемых прогнозов все цело зависит от разработчика программного обеспечения. На начальных этапах существования веб-приложения достаточным решением поставленной проблемы будет весьма тривиальных подход, предполагающий, что оптимальный прогноз для рекомендации есть не что иное, как усредненная по цене совокупность доступного множества прогнозов в базе данных.

Цель данной работы – реализовать веб-приложение, которое предоставит удобную платформу для взаимодействия двух видов пользователей: читателей и специалистов с предполагаемыми курсами акций.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1) Сформировать задание и необходимый функционал;
2) Сформировать требования к базе данных, определить круг задач, которые будут решаться с использованием созданной базы данных;
3) Спроектировать модель базы данных;
4) Спроектировать приложение для доступа к базе данных;
5) Осуществить выбор СУБД и технических средств на основе сформированных требований;
6) Создать базу данных и необходимые сущности;
7) Разработать программный интерфейс для работы с базой данных;
8) Разработать программное обеспечение для взаимодействия пользователя и базы данных.

1 Аналитическая часть

В данном разделе будет проанализирована поставленная задача и рассмотрены различные способы ее реализации.

1.1 Постановка задачи

Необходимо разработать веб-приложение, позволяющее пользователю ознакомиться с прогнозами курса акций различных специалистов, хранящимися в базе данных. Пользователь должен иметь возможность получения прогнозируемой цены одной из предложенных акций, помимо этого для удобства восприятия информации ему будет предоставлена графическая визуализация прогноза на определенном промежутке времени.

Также требуется разработать ролевую модель, на основе которой будет строиться взаимодействие с базой данных. Будут предусмотрены три роли: пользователь, специалист и администратор. Следствием ролевой модели будет необходимость создания системы регистрации и авторизации пользователей, а также возможность назначения ролей администратором. Сделать возможным для специалиста изменение и добавление прогнозов в существующую базу данных, в свою очередь для администратора доступ к базе данных будет абсолютным.

1.2 Формализация данных

База данных должна хранить информацию о:

- Пользователях;
- Прогнозах;
- Метаданных прогноза;
- Группах пользователей и их правах доступа;
- Метаданные пользователей;
- Действиях, совершаемых пользователями – журнал аудита;

Таблица 1.1 – категории и сведения о данных

Категория	**Сведения**
Пользователь	ID пользователя, ID метаданных пользователя, ID группы пользователя.
Прогноз	ID записи, код акции, ID информации об акции, дата прогноза, дата обновления прогноза, минимальная, максимальная и наиболее ожидаемая цена акции.
Метаданные прогноза	ID записи, название акции, имя источника прогноза, код акции, ссылка на подробное описание акции.
Группы пользователей	ID записи, ID группы, название группы, ID прав доступа, ID пользователя входящего в эту группу.
Права групп пользователей	ID записи, ID группы, права доступа группы.

Метаданные пользователя	ID пользователя, пароль, выбранное пользователем имя, реальное имя и фамилия пользователя, почта, дата регистрации и последней авторизации.
Журнал аудита	ID записи, действие, ID объекта над которым производилось действие, ID пользователя, совершившего действие, ID группы пользователя, информации об изменениях до и после.

1.3 Типы пользователей

Из задачи ясно, что авторизация пользователей необходима. Это делит пользователей прежде всего на авторизованных и неавторизованных.

Далее для управления приложением необходима ролевая модель: обычный пользователь, специалист, администратор.

Таблица 1.2 – типы пользователей и их функционал

Тип пользователя	**Функционал**
Гость (Неавторизованный пользователь)	Регистрация, авторизация
Авторизованный пользователь.	Выход, просмотр и выбор прогнозов, просмотр информации об акциях.
Специалист	Выход, просмотр, выбор и редактирование прогнозов, просмотр информации об акциях.
Администратор	Выход, просмотр и изменение всей информации, доступной в базе данных, в том числе права доступа групп и отдельных пользователей.

1.4 Анализ существующих решений

Рассмотрим типовые существующие решения поставленной задачи со стороны рядового пользователя.

1.4.1 Статьи аналитиков

Самым распространённым решением получения информации о предполагаемой цене и перспективах определенной акции является прочтение статьей различных аналитиков. С подобным подходом сталкивался практически любой пользователь, приводить пример таких статьей не имеет особо смысла, так как их в интернете представлено безмерное количество, лучше сразу акцентировать внимание на положительных и отрицательных аспектах такого подхода.

Положительные:

- Информация представлена в развернутом виде;
- Прогноз предполагаемой цены зачастую подкреплен аргументацией;

Отрицательные:

- Субъективность отдельно взятого автора по отношению к акции;
- Общедоступные статьи становятся вектором стратегии слишком большого количества пользователей;
- Наиболее точные серьезные источники, как правило, имеют платный доступ, зачастую достаточно дорогой;

1.4.2 Сайты с агрегацией прогнозов

Подобные сайты демонстрируют мнения различных аналитиков, зачастую используя консенсус-прогноз — это метод прогнозирования, основанный на вычислении среднего значения прогнозов различных аналитических центров (банков, инвестиционных компаний и так далее). Облик и метод подачи информации сайтов-агрегаторов разнообразен, попробуем также выделить положительные и отрицательные аспекты каждого из них.

Tinkoff Инвестиции

На рисунке 1.1 приведен интерфейс консенсус-прогноза на сайте Tinkoff

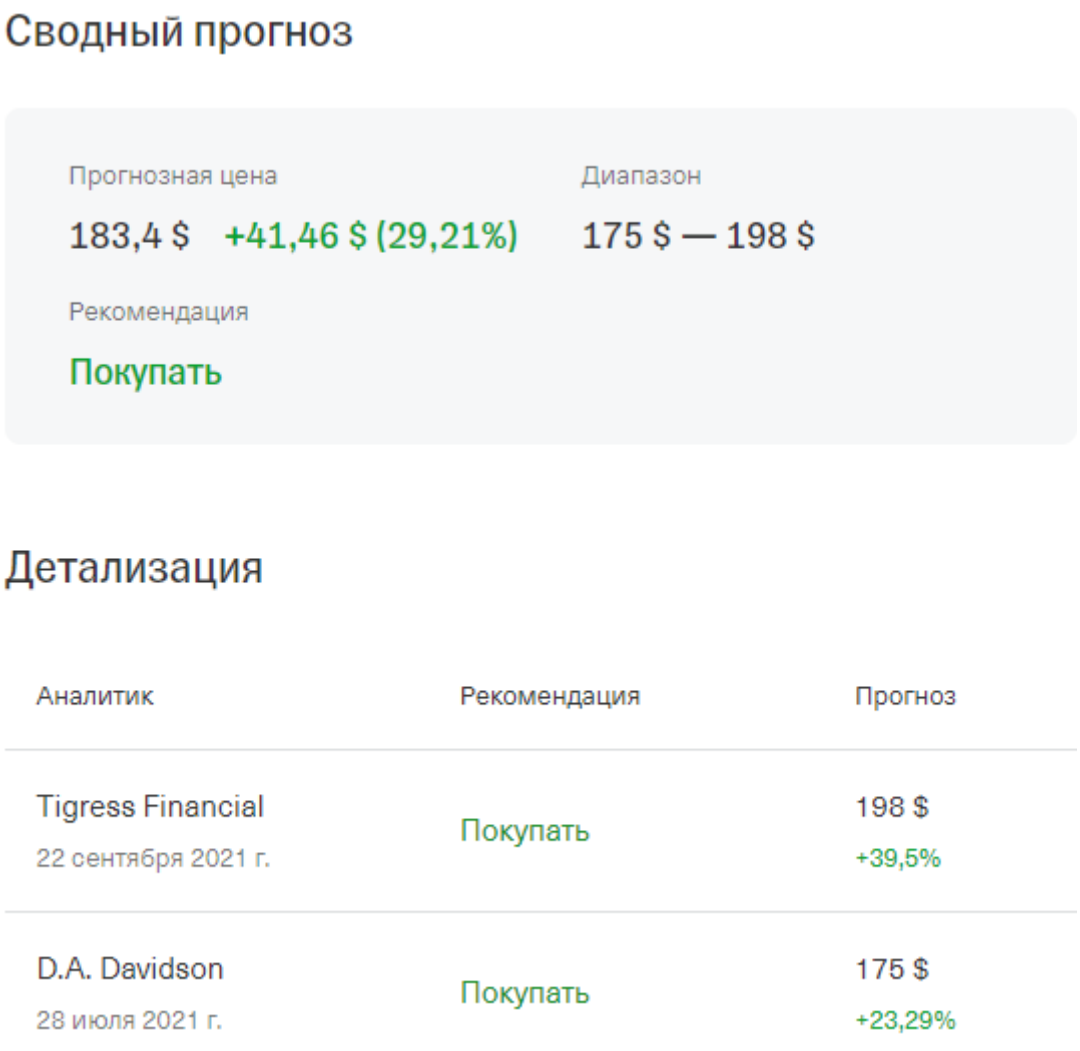

Рисунок 1.1. – консенсус-прогноз акций Apple с сайта Tinkoff Инвестиции.

Сервис предлагает возможность ознакомиться со сводным прогнозом на курс акции, в дополнение детализируя какие именно аналитики с каким прогнозом учитывались. При взаимодействии с сервисом сразу можно выделить следующие качества такого подхода к составлению консенсус-прогноза.

Положительные:

- Удобен для неопытного пользователя;
- Учитывает ожидания различных аналитиков;

Отрицательные:

- Отсутствует возможность нажать на аналитика и ознакомиться с информацией в развернутом виде;

RBC Инвестиции

На рисунке 1.2 приведен интерфейс прогнозов на сайте RBC Инвестиций.

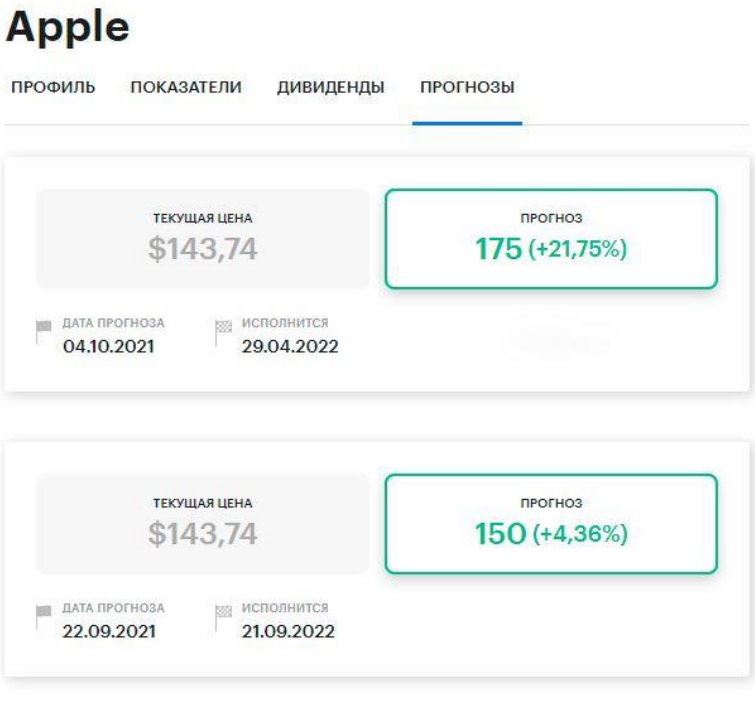

Рисунок 1.2. - прогнозы акций Apple с сайта RBC Инвестиции.

Сервис предлагает возможность ознакомиться с прогнозами различных аналитиков. При работе с сайтом можно выделить следующие аспекты.

Положительные:

- Имеются различные прогнозы аналитиков;

Отрицательные:

- Отсутствует возможность нажать на аналитика и ознакомиться с информацией в развернутом виде;
- Отсутствует какой-либо консенсус-прогноз на основе мнений аналитиков;

Investing.com

На рисунке 1.3 приведена часть интерфейса прогнозов на русскоязычной версии сайта investing.com.

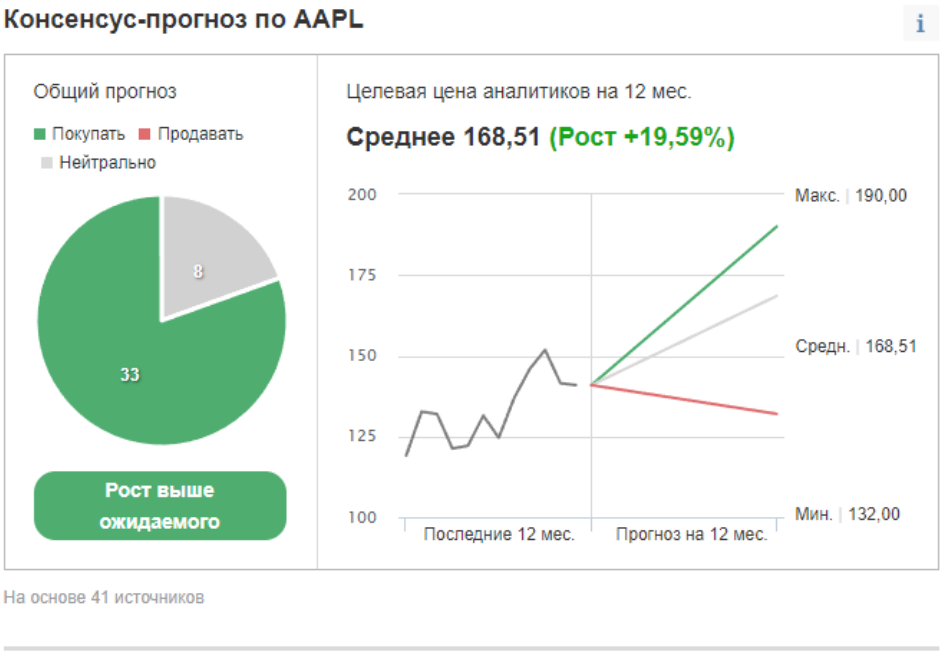

Рисунок 1.3. – прогноз акций Apple с сайта Investing.com.

Сервис предлагает огромный функционал, расписывать который можно достаточно долго, важно отметить тот факт, что доступ к расширенному перечню возможностей имеет платный доступ, ввиду этого разберем его бесплатную часть.

Положительные:

- Консенсус-прогноз имеет не только предполагаемую цену, а целый график возможных исходов;
- Обширное количество задействованных источников;

Отрицательные:

- Отсутствует какая-либо информация об используемых источниках в консенсус-прогнозе;
- Платный доступ к расширенному функционалу;

1.4.2 Вывод на основе существующих решений

На основе анализа вышеизложенного можно сделать вывод, что существующие решения имеют определенные недостатки. Актуальность темы обуславливается включением наиболее удачных решений, а именно:

- Удобство в использование для неопытного пользователя;
- Бесплатный доступ;
- Агрегация обширного количества источников;
- Возможность создания консенсус-прогноза на основе предпочтительных специалистов;

1.5. Описание существующих баз данных и систем управления базами данных

В задаче разбора и хранения информации рабочей программы важную роль играет выбранная модель хранения данных. Для персистентного хранения данных используются базы данных. Для управления базами данных используются системы управления базами данных (сокращенно СУБД). Система управления базами данных – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

1.5.1 Классификация баз данных по способу хранения

Базы данных, по способу хранения, делятся на две группы – строковые и колоночные. Каждый из этих типов служит для выполнения для определенного рода задач.

Строковые базы данных

Строковыми базами данных называются такие базы данных, записи которых в памяти представляются построчно. Строковые баз данных используются в транзакционных системах. Для таких систем характерно большое количество коротких транзакций с операциями вставки, обновления и удаления данных - INSERT, UPDATE, DELETE.

Основной упор в системах делается на очень быструю обработку запросов, поддержание целостности данных в средах с множественным доступом и эффективность, которая измеряется количеством транзакций в секунду. Схемой, используемой для хранения транзакционных баз данных, является модель сущностей, которая включает в себя запросы, обращающиеся к отдельным записям. Так же, в системах есть подробные и текущие данных.

Колоночные базы данных

Колоночными базами данных называются базы данных, записи которых в памяти представляются по столбцам. Колоночные базы данных используется в аналитических системах. Характеризуется низким объемом транзакций, а запросы часто сложны и включают в себя агрегацию. Время отклика для таких систем является мерой эффективности. Подобные системы широко используются методами интеллектуального анализа данных. В таких базах есть агрегированные, исторические данные, хранящиеся в многомерных схемах.

1.5.2 Выбор модели хранения данных

Для решения задачи построчное хранение данных преобладает над колоночным хранением по нескольким причинам:

- Задача предполагает постоянное добавление и изменение данных;
- Задача предполагает быструю отзывчивость на запросы пользователя;

1.5.3 Обзор СУБД с построчным хранением

В настоящее время существует множество различных СУБД, наиболее популярные из которых: Oracle, MySQL, PostgreSQL. В связи с выбором реляционной модели данных, необходимо провести сравнительный анализ между наиболее популярными реляционными СУБД, такими как PostgreSQL, MS SQL Server и Oracle.

Oracle

Oracle Database — это объектно-реляционная СУБД (система управления базами данных), созданная компанией Oracle.

Все транзакции Oracle Database соответствуют обладают свойствами ACID, поддерживает триггеры, внешние ключи и хранимые процедуры. Данная СУБД подходит для разнообразных рабочих нагрузок и может использоваться практически в любых задачах. Особенностью Oracle Database является быстрая работа с большими массивами данных.

Достоинства:

- Поддержка огромных баз данных и большого числа пользователей;
- Быстрая обработка транзакций;

Недостатки:

- Требует значительные вычислительные ресурсы;

MySQL

MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle. Если сравнивать MySQL с другими СУБД с открытым кодом, то главное отличие заключается в бесперебойной работе с интерфейсом API. Это ПО позволяет любому пользователю получить доступ к системе управления БД, даже если для написания был использован любой другой язык программирования.

Достоинства:

- Прост в использовании;
- Может работать с другими базами данных;
- Поддерживает основную часть функционала SQL;

Недостатки:

- Ограниченная функциональность;

PostgreSQL

PostgreSQL – это свободно распространяемая объектно-реляционная система управления базами данных, наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

PostgreSQL предоставляет транзакции со свойствами атомарности, согласованности, изоляции, долговечности, автоматически обновляемые представления, материализованные представления, триггеры, внешние ключи и хранимые процедуры. Данная СУБД предназначена для обработки ряда рабочих нагрузок, от отдельных компьютеров до хранилищ данных или веб-сервисов с множеством одновременных пользователей.

Достоинства:

- Полный SQL функционал
- Масштабируем, имеет способность к обработке огромного количества данных. Расширяем за счет использования хранимых процедур

Недостатки:

- Скорость работы уменьшается во время пакетных операций или запросов на чтение.

1.6 Выводы из аналитической части

В данном разделе была проанализирована поставленная задача и способы ее реализации, обоснована актуальность темы, рассмотрены существующие базы данных, а также системы управления базами данных.

2. Конструкторская часть

В следующем разделе будут формализованы сущности, правила и прецеденты системы, описаны спроектированная база данных и приложение.

2.1 Формализация сущностей системы

На рисунке 2.1. отображена диаграмма сущностей системы. Данная схема построена на основе категории и сведений о данных, отображенных в таблице под номером 1.1.

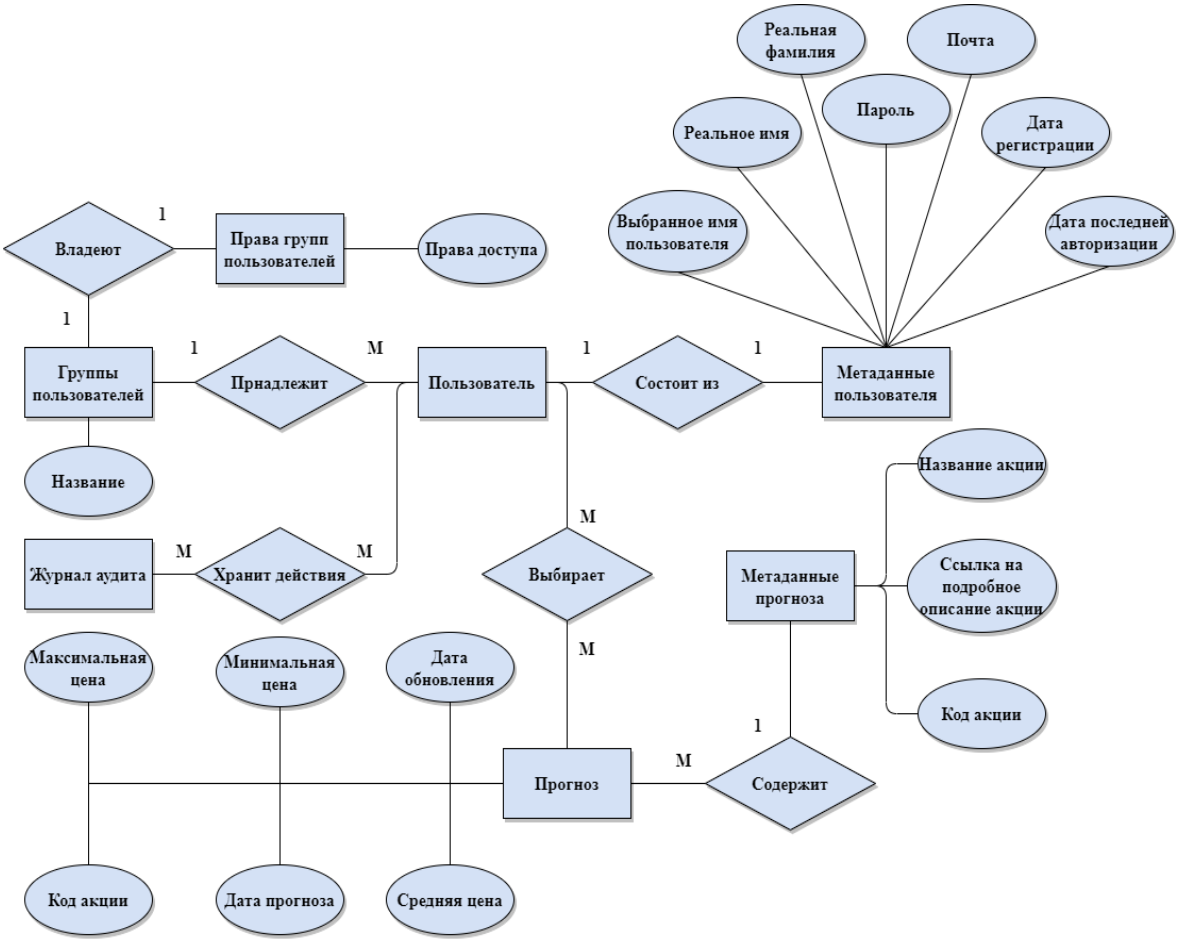

Рис 2.1 – диаграмма сущностей.

2.2 Функциональная модель

На рисунке 2.2. изображена функциональная модель, отображающая структуру и функции системы.

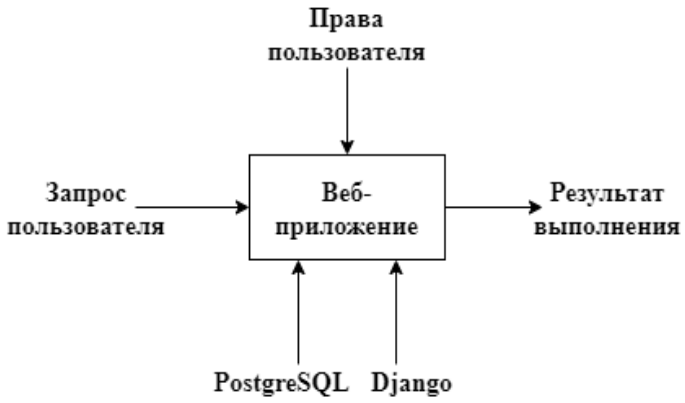

Рис. 2.2 – функциональная модель.

2.3 Сценарий использования

В данном разделе необходимо построить Use Case Diagram (диаграмму прецедентов). Она состоит из графической диаграммы, описывающей действующие лица и прецеденты – конкретные действия, которые выполняет пользователь при работе с системой. Отображенные диаграммы являются расширением таблицы 1.2.

На рисунке 2.3 предоставлена диаграмма сценариев использования для неавторизованного пользователя.

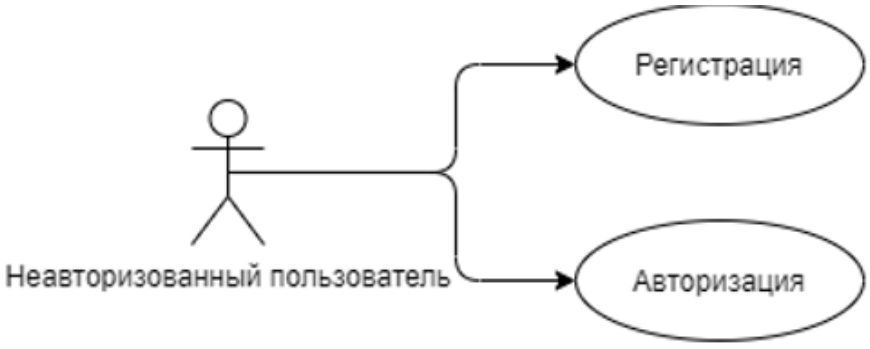

Рис. 2.3 – Диаграмма сценариев использования для неавторизованного пользователя.

На рисунке 2.4 предоставлена диаграмма сценариев использования для авторизованного пользователя.

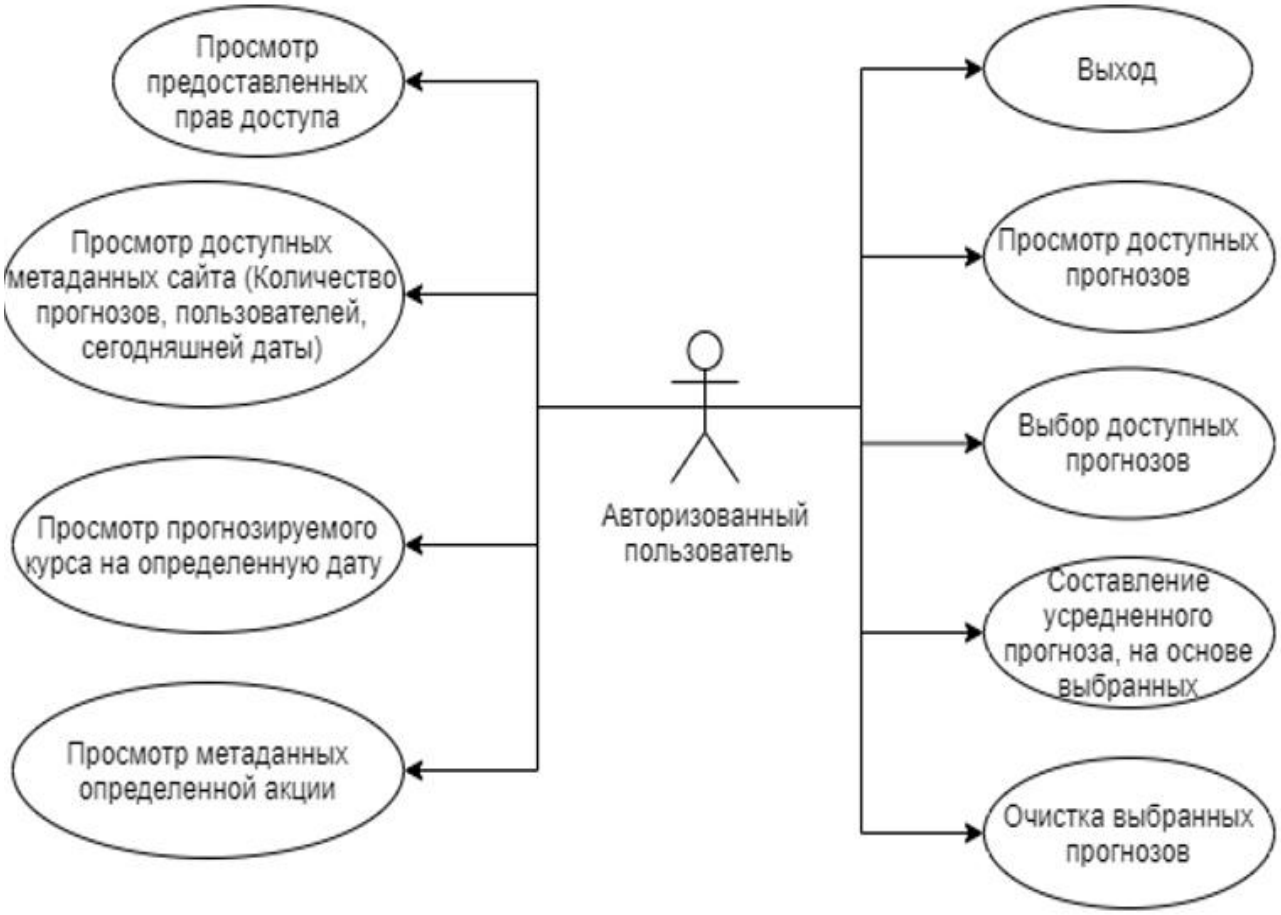

Рис. 2.4 – Диаграмма сценариев использования для авторизованного пользователя.

На рисунке 2.5 предоставлена диаграмма сценариев использования для специалиста.

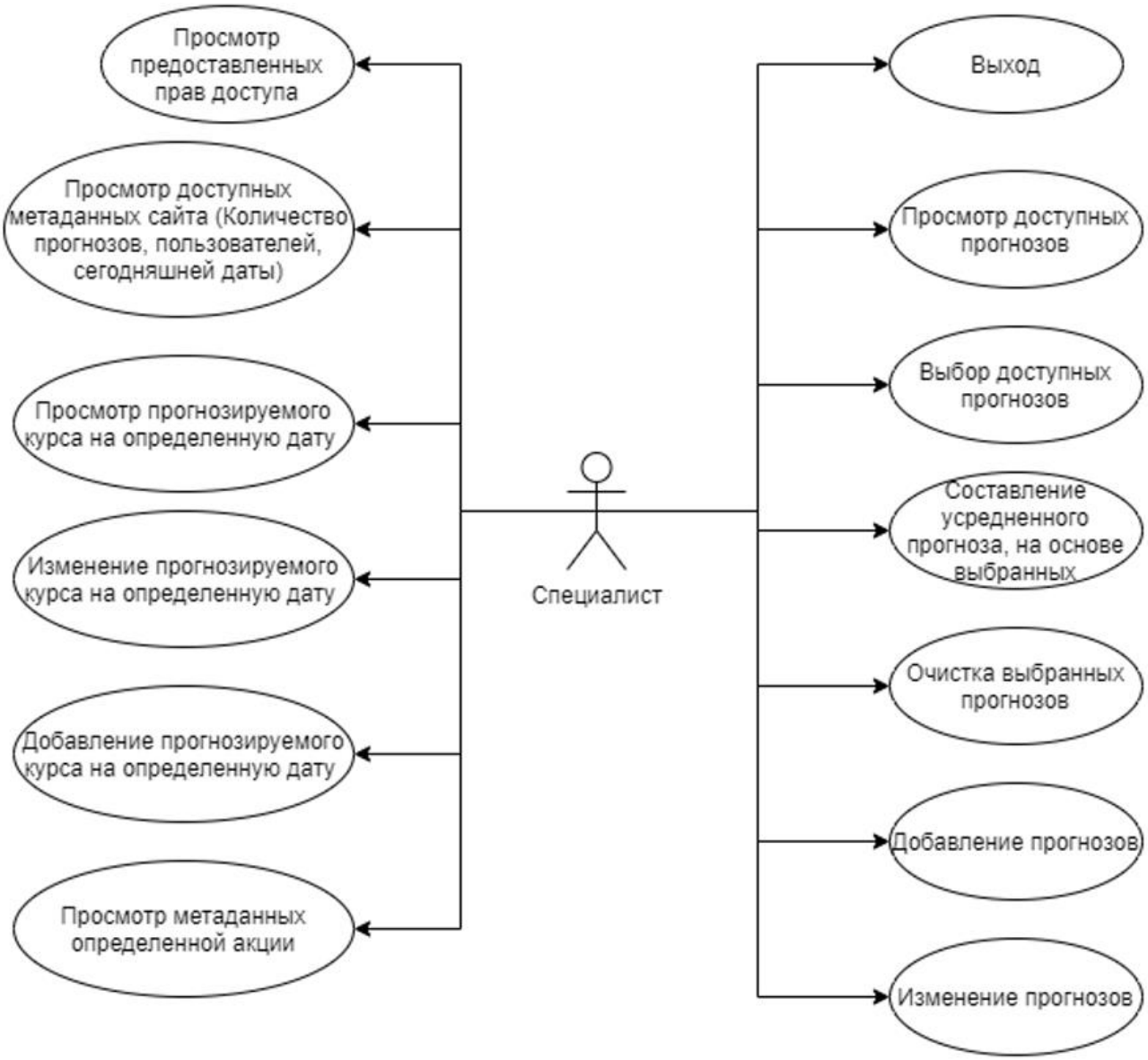

Рис. 2.5 – Диаграмма сценариев использования для специалиста.

На рисунке 2.6 предоставлена диаграмма сценариев использования для администратора.

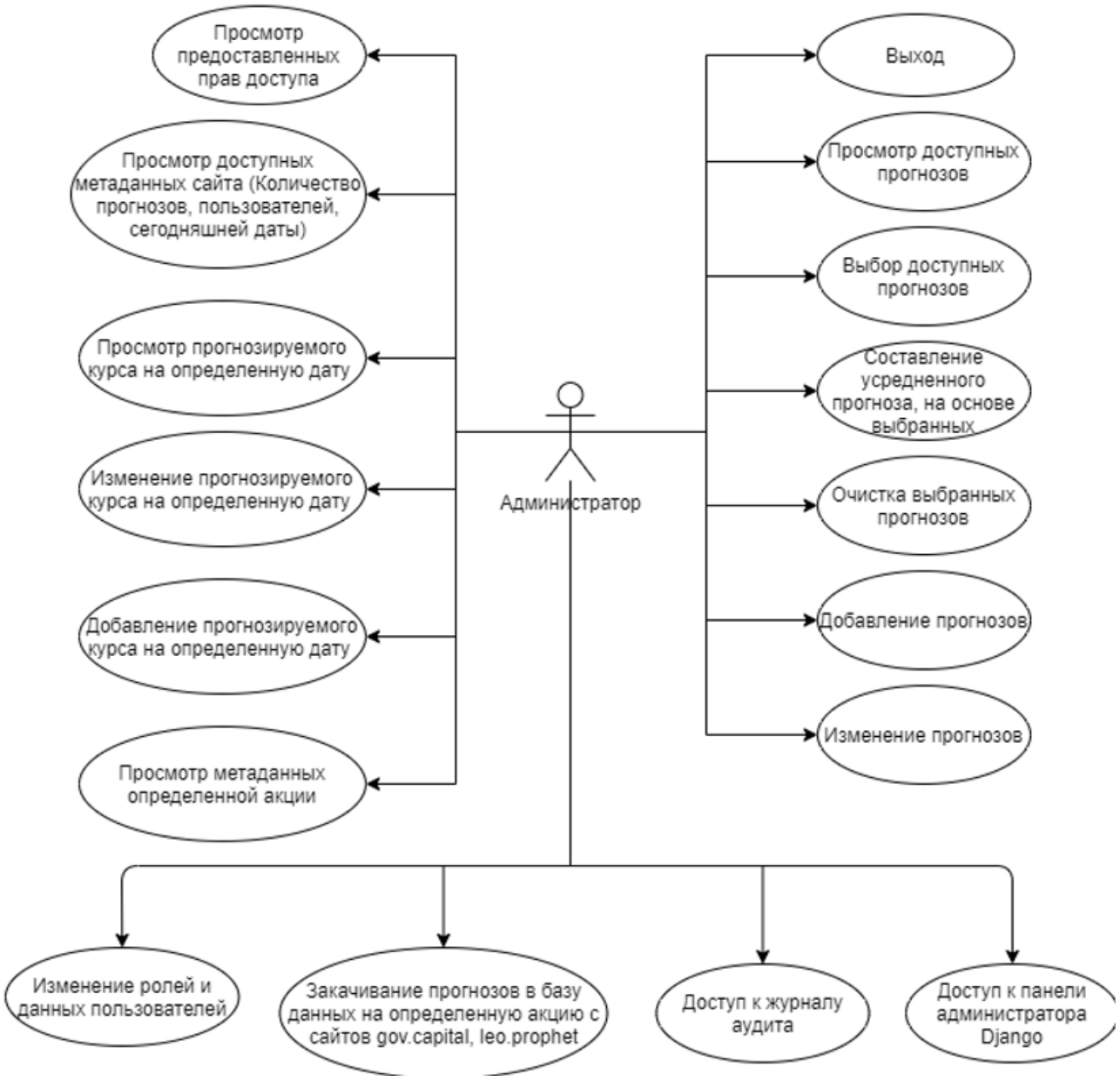

Рис. 2.6 – Диаграмма сценариев использования для администратора.

2.4 Проектирование базы данных

База данных должна хранить рассмотренные в таблице 1.1 данные. В соответствии с этой таблицей можно выделить следующие таблицы:

- таблица авторизованных пользователей auth_user;
- таблица групп авторизованных пользователей auth_user_groups;
- таблица прав групп авторизованных пользователей auth_group_permissions;
- таблица прав авторизованных пользователей auth_user_permissions;
- таблица прав доступа auth_persmission;
- таблица моделей базы данных content_type;
- таблица прогнозов на определенную дату order;
- таблица метаданных акции stockinfo;
- таблица журнала аудита admin_log;

Таблица **auth_user** должна хранить информацию об авторизованном пользователе:

- id – уникальный идентификатор пользователя, PK, id;
- username – имя выбранное пользователем, varchar;
- password – пароль пользователя, int;
- email – почтовый ящик пользователя, varchar;
- date_joined – дата регистрации пользователя, date;
- last_login – дата последней авторизации пользователя, date;

Таблица **auth_user_groups** должна хранить о принадлежности пользователей каким-либо группам:

- id – уникальный идентификатор записи, PK, id;
- user_id – уникальный идентификатор пользователя, FK, id;

- group_id – уникальный идентификатор группы, FK, id;

Таблица **auth_group_permissions** должна хранить какие права доступа принадлежат группе:

- id – уникальный идентификатор записи, PK, id;
- group_id – уникальный идентификатор группы пользователей, FK, id;
- permission_id – уникальный идентификатор прав доступа, FK, id;

Таблица **auth_user_permissions** должна хранить какие права доступа принадлежат пользователю:

- id – уникальный идентификатор записи, PK, id;
- user_id – уникальный идентификатор пользователя, FK, id;
- permission_id – уникальный идентификатор прав доступа, FK, id;

Таблица **auth_permission** должна хранить права доступа:

- id – уникальный идентификатор записи, PK, id;
- name – название прав доступа, varchar;
- content_type_id – уникальный идентификатор модели БД, FK, id;
- codename – имя кода выполнения, varchar;

Таблица моделей базы данных **content_type**:

- id - уникальный идентификатор записи, PK, id;
- model – название модели, varchar;

Таблица admin_log должна хранить журнал аудита сайта:

- id - уникальный идентификатор записи, PK, id;
- action_time – время совершения действия, date;

- object_id – имя объекта, над которым совершалось действие, varchar;
- action_flag – совершенное действие, varchar;
- change_message – сообщение об совершенном изменение, varcar;
- content_type_id – уникальный идентификатор модели БД, FK, id;
- user_id – уникальный идентификатор авторизованного пользователя, FK, id;

Таблица **stockinfo** содержащая метаданные об акциях:

- id - уникальный идентификатор записи, PK, id;
- name – название акции, varchar;
- ticket – код акции, varchar;
- source – имя специалиста или источника давшего прогноз

Таблица **order** содержащая прогноз на определенную дату:

- id - уникальный идентификатор прогноза, PK, id;
- product_name – название акции, varchar;
- price – наиболее вероятная или средняя цена акции, int;
- min_price – минимальная возможная цена акции, int;
- max_price – максимально возможная цена акции, int;
- created_time – время создания прогноза, date;
- updated_time -время обновления прогноза, date;
- ticket_id – уникальный идентификатор метаданных акции, FK, int;

На рисунке 2.7 предоставлена диаграмма реализованной базы данных.

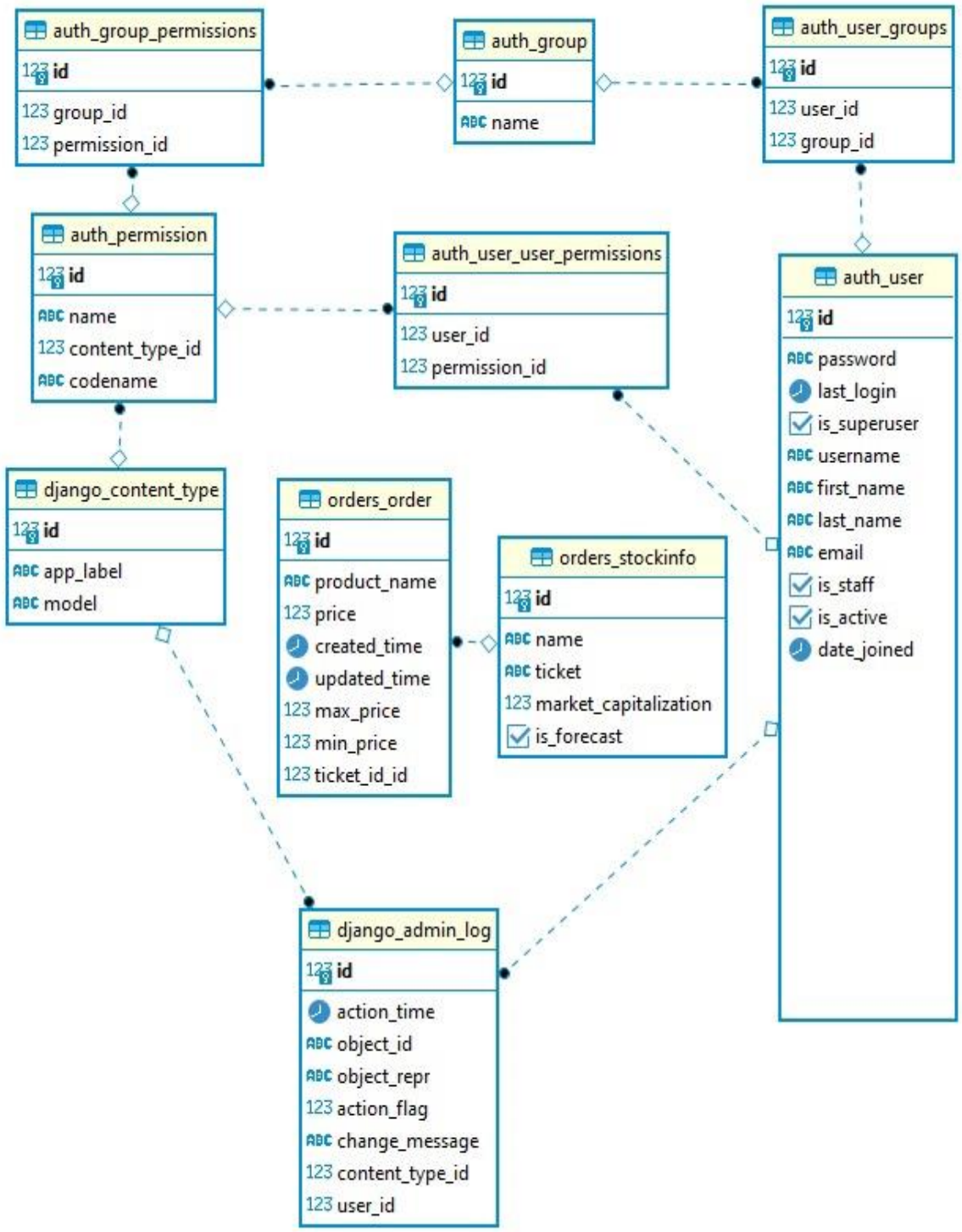

Рис. 2.7 – диаграмма базы данных.

2.5 Проектирование приложения

Приложение для работы с базой данных представляет собой многокомпонентное веб-приложение. Выделены компонент доступа к данным и компонент бизнес-логики. Компонент бизнес-логики спроектирован по схеме MVC. Программа будет состоять из двух составных частей:

1) front-end – приложение, с которым непосредственно взаимодействует пользователь;

2) back-end – приложение, с которым взаимодействует front-end и база данных: получение, изменение, удаление данных

Для корректной работы проекта back-end часть приложения будет состоять из следующих составных частей: API, database, parser.

API (Application Programming Interface) – серверная часть приложения, которая должна принимать, обрабатывает запросы от front-enda взаимодействует с промежуточным слоем между API и database.

Database – часть приложения, которая должна содержать модели для работы с базой данных, а также клиент, который принимает запросы от API и отправляет их непосредственно базе данных.

Parser – часть приложения, которая должна получать, обрабатывать необходимые данные для работы агрегатора. Реализация parser’а зависит от конкретного сайта, с которого собираются данные, но схематически в общем виде его алгоритм можно представить рисунком 2.5.1.

Рисунок 2.5.1 – Общий алгоритм parser’а данных с сайтов.

Для удовлетворения требованиям к программе, необходимы следующие функции доступа к данным:

- get_user_by_email, get_user_by_id – функция получения данных пользователя, если он существует;
- delete_user_by_id – функция удаления пользователя;
- grant_user_specialist_by_id – функция добавления роли “Специалист” пользователю;
- grant_user_admin_by_id – функция добавления роли “Администратор” пользователю;

- ungrant_user_specialist_by_id – функция удаления роли “Специалист” пользователя;
- ungrant_user_admin_by_id – функция удаления роли “Администратор” пользователю;
- update_user – функция для обновления прав доступа пользователю, то есть заполнения данных о нем в других таблицах;
- outdated_data_removing – удаления неактуальных прогнозов по дате;

Также для работы с таблицей необходимо реализовать триггеры:

- outdated_data_remove_trigger – триггер, который вызывает функцию outdated_data_removing при обновление таблицы order_orders;
- update_user_trigger – триггер, который вызывает функцию update_user после функций изменения роли пользователя;

Специалист и администратор смогут взаимодействовать с базой данных при помощи интерфейса сайта, а также панели администрирования, при необходимости они также смогут совершать запросы напрямую в базе данных.

2.6 Выводы из конструкторской части

В данном разделе были формализованы сущности, правила и прецеденты системы, выбрана модель базы данных, описаны спроектированные
база данных и приложение.

3. Технологическая часть

В следующем разделе будут выбраны СУБД и технологический стек для разработки системы. Приведена реализация компонентов доступа к базе данных и бизнес-логики, описан презентационный уровень приложения.

3.1 Выбор и обоснование инструментов разработки

Для выполнения проекта был выбран язык программирования Python по следующим причинам:

- Кроссплатформенность. Python – это интерпретируемый язык, его интерпретаторы существуют для многих платформ. Поэтому с запуском его на любой ОС не должно возникнуть проблем;
- С Python доступно огромное количество сервисов, сред разработки, и фреймворков. Легко можно найти подходящий продукт для работы;
- Возможность подключить библиотеки, написанные на C. Это позволяет повысить эффективность, улучшить быстродействие;
- Наличие самых разных источников информации о Python. Не составит труда найти ответ на любой возникший вопрос, так существует много бесплатной литературы, обучающих видео-пособий, готовых исходников и шаблонов для работы в открытом доступе;

Для решения задачи была выбрана СУБД PostgreSQL, потому что данная СУБД обеспечивает целостность данных, поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных. Кроме того, PostgreSQL проста в развертывании.

Приложение разрабатывалось с использованием веб-фреймворка Django. Фреймворк реализует схему MVC и предоставляет ORM инструменты для работы с базой данных, кроме того, позволяет выполнять чистые SQL запросы.

Front-end часть веб-приложения разрабатывалась с использованием с

использованием следующих языков: Html, CSS, JavaScript, в которые входили следующие наборы и подходы: Bootstrap, jQuery, AJAX.

В качестве среды разработки было выбрано программное обеспечение IntelliJ IDEA от JetBrains из-за следующих причин:

- Бесплатна в пользовании для студентов;
- Поддерживает многие языки программирования, в частности Python и JavaScript;
- Имеет удобные короткие команды, что значительно ускоряет разработку;
- Имеет возможность отладки кода;
- Имеет множество плагинов для удобной разработки приложения;

3.2 Детали реализации

В листинге 3.2.1 представлена модель таблицы StockInfo, которая отвечает за метаданные акции.

Листинг 3.2.1. – модель таблицы StockInfo.

```
class StockInfo(models.Model):
    name = models.TextField(null=True)
    ticket = models.TextField(max_length=5)
    source = models.TextField(null=True)

    class Meta:
        ordering = ['ticket']
```

В листинге 3.2.2. представлена модель таблицы Order, которая отвечает за конкретные прогнозы акций на определенную дату.

Листинг 3.2.2. – модель таблицы Order.

```
class Order(models.Model):
    MONTHS = {1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr', 5: 'May', 6:
'Jun', 7: 'Jul', 8: 'Aug', 9: 'Sep', 10: 'Oct',
              11: 'Nov', 12: 'Dec'}
    product_name = models.CharField(max_length=5)
    ticket_id = models.ForeignKey(StockInfo,
related_name='stock_info_id',
```

```
                                    on_delete=models.SET_NULL,
null=True)
    price = models.IntegerField()
    min_price = models.IntegerField()
    max_price = models.IntegerField()
    created_time = models.DateTimeField(db_index=True)
    updated_time = models.DateTimeField(auto_now=True)

    class Meta:
        verbose_name = 'order'
        verbose_name_plural = 'orders'
```

В листинге 3.2.3 представлен метод таблицы Order под названием orders_month_report, который позволяет сформировать запрос к базе данных, который возвращает среднюю прогнозируемую цену акции на следующий год по месяцам.

Листинг 3.2.3. – метод orders_month_report.

```
@classmethod
def orders_month_report(cls):
    now = datetime.datetime.now()

    filter_params = {
        'created_time__date__gte': now.date(),
        'created_time__date__lte': '{year}-{month}-
{day}'.format(year=now.year + 1, month=now.month,

day=now.day),
    }

    annotate_params = {
        'total_order': Count('id'),
        'total_price': Avg('price'),
    }
    queryset =
cls.objects.filter(**filter_params).values('created_time__year',
'created_time__month').annotate(
        **annotate_params)

    return queryset, [cls.MONTHS[data.get('created_time__month')]
for data in queryset]
```

В листинге 3.2.4. представлена форма модели Order, необходимая для взаимодействия пользователей с таблицей.

Листинг 3.2.4. – форма модели Order.

```
class OrderForm(forms.ModelForm):
    product_name = forms.CharField(label="Ticket",
widget=forms.TextInput(attrs={'class': 'form-control order'}))
    min_price = forms.IntegerField(label="Min Price",
widget=forms.TextInput(attrs={'class': 'form-control order'}))
    price = forms.IntegerField(label="Average Price",
widget=forms.TextInput(attrs={'class': 'form-control order'}))
    max_price = forms.IntegerField(label="Max Price",
widget=forms.TextInput(attrs={'class': 'form-control order'}))
    created_time = forms.DateTimeField(label="Date",
widget=forms.TextInput(
        attrs={'class': 'form-control order', 'placeholder': 'YY-
mm-dd H:i:s'}))

    class Meta:
        model = Order
        fields = ['product_name', 'min_price', 'price',
'max_price', 'created_time']
```

В листинге 3.2.5. представлен заполнения содержания стартовой страницы с помощью различных запросов к базе данных:

Листинг 3.2.5. – Заполнение стартовой страницы.

```
@login_required(login_url="/login/")
def index(request):
    global array_of_id
    context = {'segment': 'index'}
    html_template = loader.get_template('dashboard.html')
    context['today_date'] = datetime.date(datetime.now())
    context['total_users'] = User.objects.all().count()
    context['current_role'] = user_role(request.user)
    context['user_role_check'] = user_role_check(request.user)
    context['total_tickets'] = StockInfo.objects.all().count()
    context['tickets'], context['info'] = set_pagination(request,
StockInfo.objects.all().order_by('-id'), item_numer=8)
    orders_month_report, orders_month_report_labels =
Order.orders_month_report()
    orders_month_report =
orders_month_report.filter(ticket_id__in=array_of_id)
    context['orders_month_report'] = list(orders_month_report)
    context['orders_month_report_labels'] =
orders_month_report_labels
```

```
    context['orders'], context['info'] = set_pagination(request,
Order.objects.filter(ticket_id__in=array_of_id).order_by('-id'),
item_numer=10)
    return HttpResponse(html_template.render(context, request))
```

В листинге 3.2.6. представлено добавление нового прогноза с сайтов gov.capital и leoprophet по коду акции.

Листинг 3.2.6. – добавление новых прогнозов в базу данных.

```
@login_required(login_url="/login/")
def ticket_add(request):
    if not request.user.is_superuser:
        context = {}
        html_template = loader.get_template('page-403.html')
        return HttpResponse(html_template.render(context,
request))
    if request.method == 'POST':
        form = AddTicketForm(request.POST)
        if form.is_valid():
            print("Adding ticket: " +
request.POST['addTicketField'])
            ticket = request.POST['addTicketField']
            name = get_name_by_ticket(ticket)
            data = gov_capital(ticket)
            stock_info = StockInfo.objects.create(name=name,
ticket=ticket, source='Gov-Capital')
            stock_info.save()
            ticket_id = stock_info.id
            i = 0
            while i < len(data):
Order.objects.create(ticket_id=StockInfo.objects.get(pk=ticket_id)
, created_time=data[i],
                                     price=data[i + 1],
min_price=data[i + 2], max_price=data[i + 3],
                                     updated_time=data[i],
product_name=ticket)
                i = i + 4

            print("Leo Prophet start")
            data = leo_prophet(ticket)
            stock_info = StockInfo.objects.create(name=name,
ticket=ticket, source='Leo-Prophet')
            stock_info.save()
            ticket_id = stock_info.id
            i = 0
            while i < len(data):
```

```
Order.objects.create(ticket_id=StockInfo.objects.get(pk=ticket_id)
, created_time=data[i],
                                        price=data[i + 1],
min_price=data[i + 2], max_price=data[i + 3],
                                        updated_time=data[i],
product_name=ticket)
                i = i + 4

    form = AddTicketForm()

    return render(request, 'addticket.html', {'form': form})
```

В листинге 3.2.7 приведен parser данных с сайта gov.capital при помощи библиотеки BeautifulSoup.

Листинг 3.2.7. – парсер данных сайта gov.capital.

```
def gov_capital(ticket):
    url = 'https://gov.capital/stock/' + ticket + '-stock/'
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'lxml')
    raw_data = soup.find_all('td', {'class': ['text-align-right',
'text-align-left']})
    forecast_date_data = []
    i = 0
    for data in raw_data:
        i = i + 1
        if i == 1:
            forecast_date_data.append(date_conversion(data.text))
            continue
        forecast_date_data.append(float_conversion(data.text))
        if i == 4:
            i = 0
    return forecast_date_data
```

В листинге 3.2.8 приведено взаимодействие с прогнозами в базе данных с помощью запросов.

Листинг 3.2.8 – взаимодействие с прогнозами в базе данных.

```
@method_decorator(login_required(login_url='login'),
name='dispatch')
class OrderView(View):
    context = {}

    def get(self, request, pk=None, action=None):
        if request.is_ajax():
            self.context['template'] = self.get_create_form(pk)
```

```
        return JsonResponse(self.context)

    def post(self, request, pk=None, action=None):
        form = OrderForm(request.POST)
        if form.is_valid():
            order = form.save()
            item = render_to_string('orders/row_item.html',
{'order': order})

            response = {'valid': 'success', 'message': 'Stock row
created successfully.', 'item': item}
        else:
            response = {'valid': 'error', 'message':
form_validation_error(form)}
        return JsonResponse(response)

    def put(self, request, pk=None, action=None):
        order = self.get_object(pk)
        form = OrderForm(QueryDict(request.body), instance=order)
        if form.is_valid():
            order = form.save()
            item = render_to_string('orders/row_item.html',
{'order': order})

            response = {'valid': 'success', 'message': 'Stock row
updated successfully.', 'item': item}
        else:
            response = {'valid': 'error', 'message':
form_validation_error(form)}

        return JsonResponse(response)

    def get_create_form(self, pk=None):
        form = OrderForm()
        if pk:
            form = OrderForm(instance=self.get_object(pk))
        return render_to_string('orders/modal_form.html', {'form':
form})

    def get_object(self, pk):
        order = Order.objects.get(id=pk)
        return order
```

В листинге 3.2.9 приведен пример триггера базы данных/

Листинг 3.2.9 – триггер удаления неактуальных прогнозов из базы данных.

```
CREATE OR REPLACE FUNCTION today_datetime()
RETURNS timestamp AS $$
    from datetime import datetime
    plan = plpy.prepare('select $1 as my_datetime', ['timestamp'])
    rv = plpy.execute(plan, (datetime.now(),))
    return rv[0]['my_datetime']
$$ LANGUAGE plpython3u;

CREATE OR REPLACE FUNCTION outdated_data_removing()
     RETURNS TRIGGER
    LANGUAGE plpgsql
AS
$$
BEGIN
    DELETE FROM orders_order
     WHERE created_time < today_datetime();
END;
$$;

CREATE TRIGGER outdated_data_remove_trigger
BEFORE UPDATE ON orders_order
EXECUTE PROCEDURE outdated_data_removing()
```

Ролевая модель

Ролевая модель реализована на нескольких уровнях. На уровне базы данных, которая продемонстрирована в листинге 3.2.10.

Листинг 3.2.10. – ролевая модель на уровне базы данных.

```
CREATE ROLE specialist WITH LOGIN PASSWORD 'specialist_password';
GRANT INSERT on orders_order to specialist;
GRANT SELECT on orders_order to specialist;
GRANT UPDATE on orders_order to specialist;
GRANT DELETE on order_order to specialist;
```

```
CREATE ROLE administration WITH LOGIN PASSWORD 'admin_password';
GRANT INSERT on orders_order to administration;
GRANT SELECT on orders_order to administration;
GRANT UPDATE on orders_order to administration;
GRANT DELETE on order_order to administration;
GRANT INSERT on auth_user to administration;
GRANT SELECT on auth_user to administration;
GRANT UPDATE on auth_user to administration;
GRANT DELETE on auth_user to administration;
GRANT INSERT on auth_user_groups to administration;
GRANT SELECT on auth_user_groups to administration;
GRANT UPDATE on auth_user_groups to administration;
GRANT DELETE on auth_user_groups to administration;
GRANT INSERT on orders_stockinfo to administration;
GRANT SELECT on orders_stockinfo to administration;
GRANT UPDATE on orders_stockinfo to administration;
GRANT DELETE on orders_stockinfo to administration;
...
```

На уровне панели администрирования Django, рисунок 3.2.11.

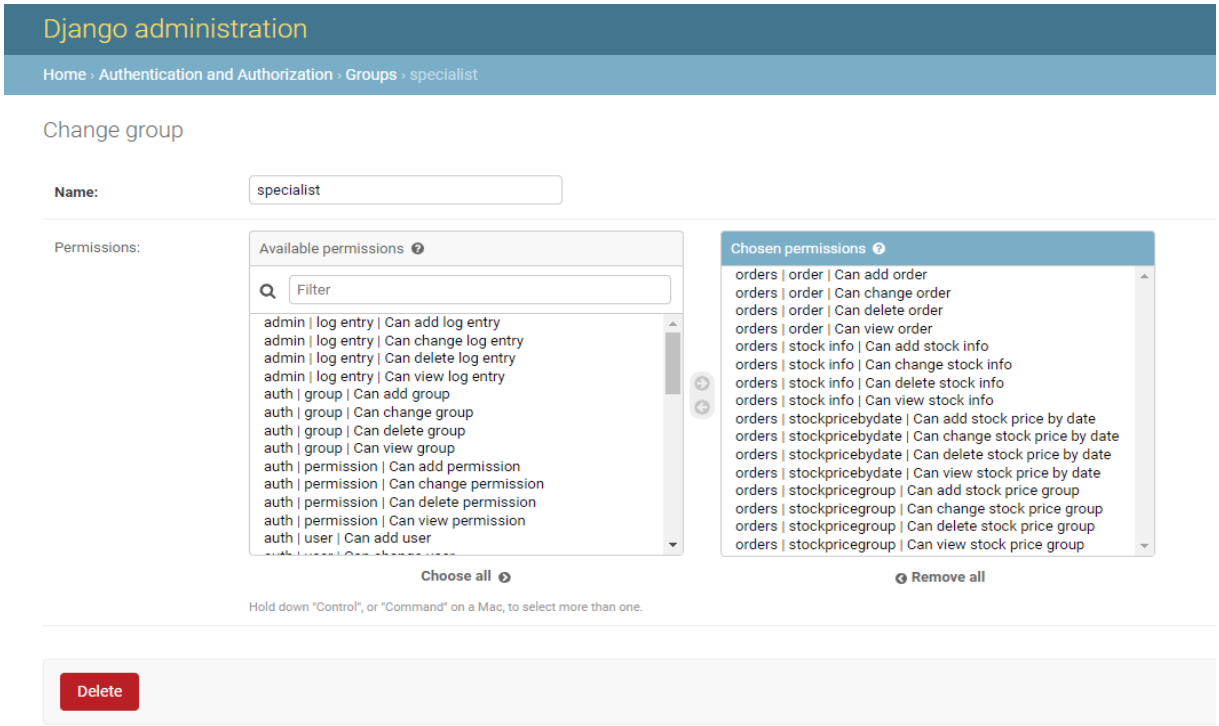

Рисунок 3.2.11. Права доступа группы специалист внутри панели администрирования Django.

На презентационном уровне идет проверка роли и прав доступа пользователя при рендере страниц, подробнее этот функционал разбирается в следующем пункте, пример его реализации можно увидеть в листинге 3.2.12.

Листинг 3.2.10. – Проверка роли пользователя при рендере элементов страницы.

```
{% if user_role_check %}
<a href="{% url 'orders' order.id %}" class="get-order-
form">Edit</a>
{% endif %}
```

3.3 Презентационный уровень приложения

Первоначально при заходе на сайт пользователю предлагают авторизироваться, либо зарегистрироваться, так как без авторизации он не получит до ступ к контенту. Страницы авторизации и регистрации показаны на рисунке 3.3.1 и 3.3.2.

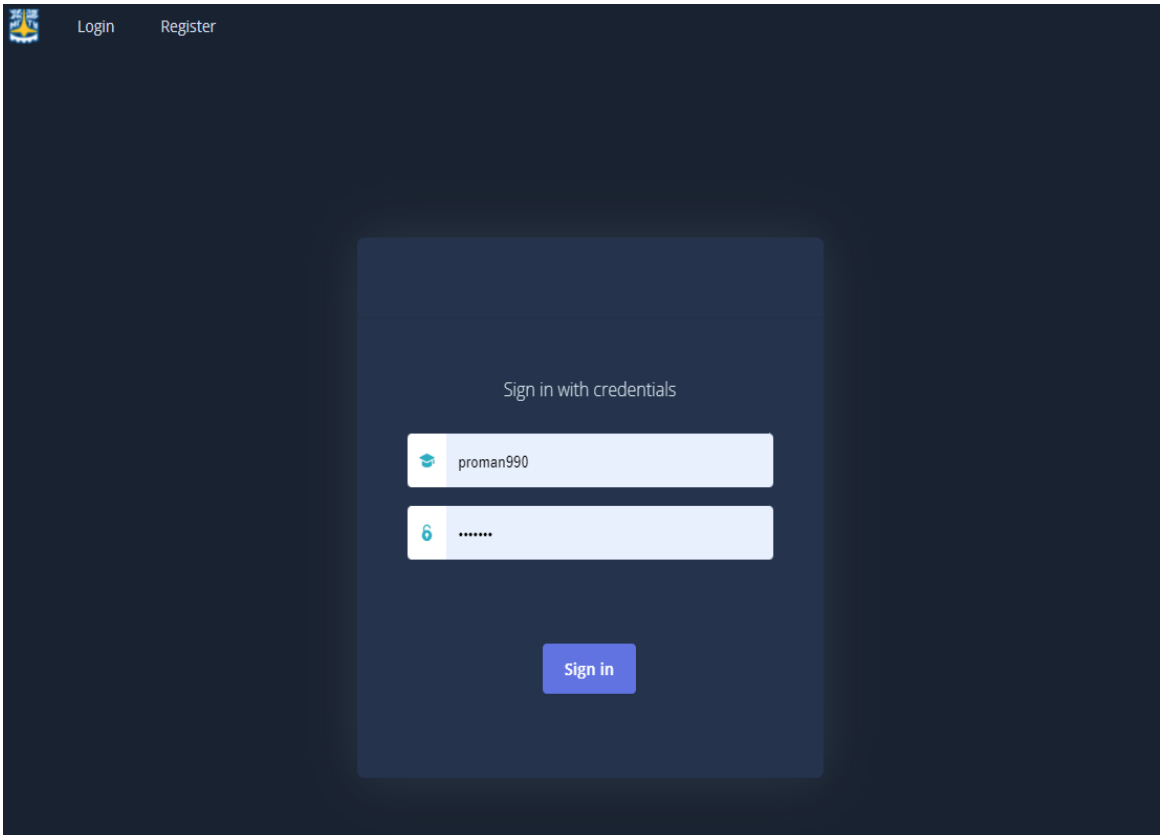

Рисунок. 3.3.1 – страница авторизации пользователя.

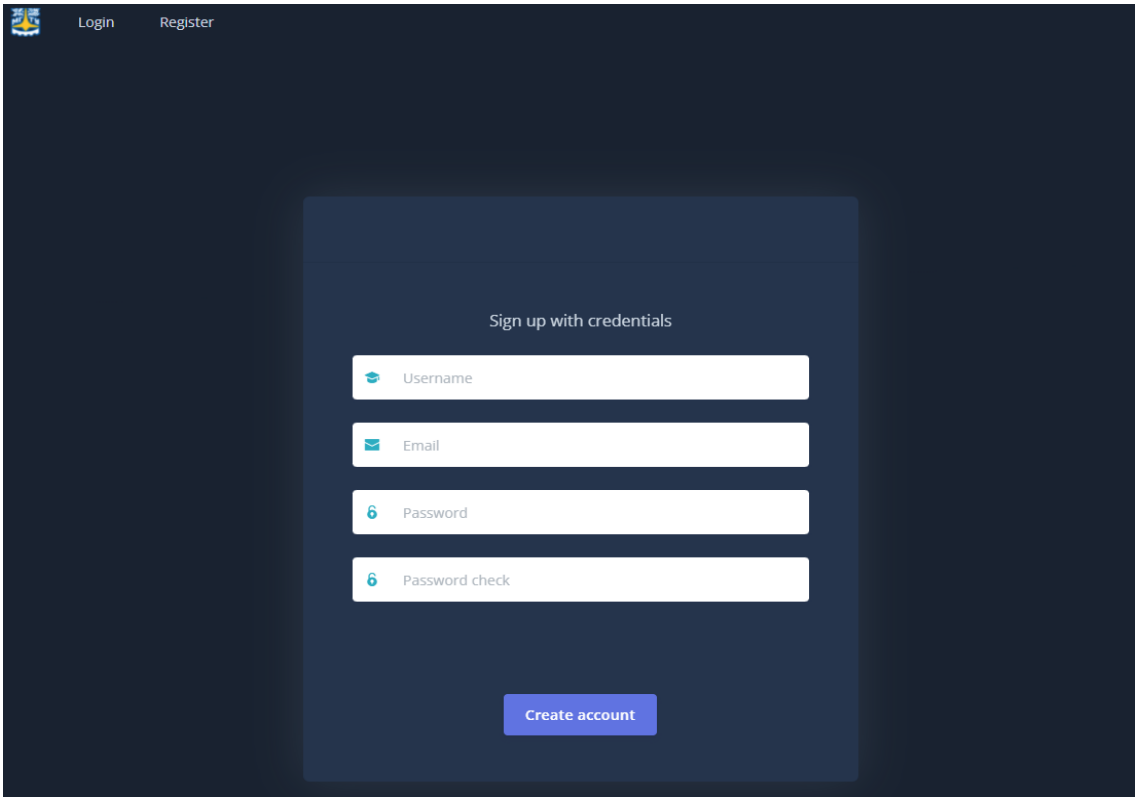

Рисунок 3.3.2 – страница регистрации пользователя.

На рисунках 3.3.3, 3.3.4 и 3.3.5. отображена главная страница сайта для пользователей различных уровней доступа.

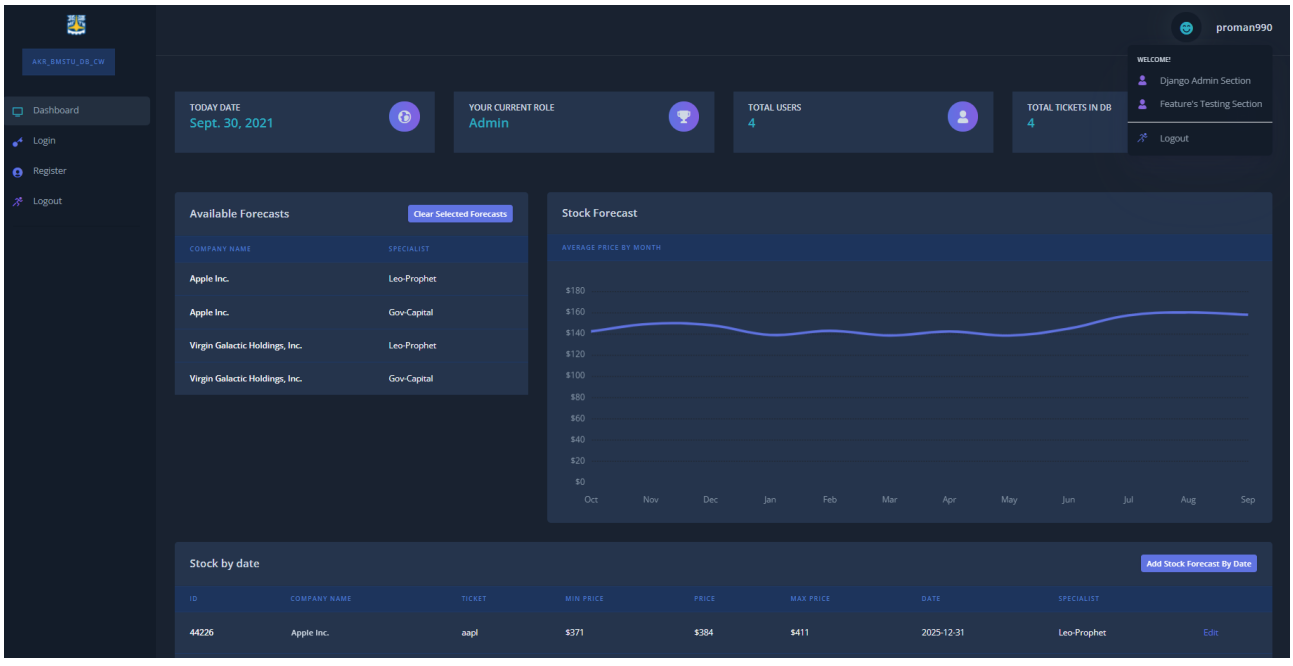

Рисунок 3.3.3 – главная страница сайта для администратора.

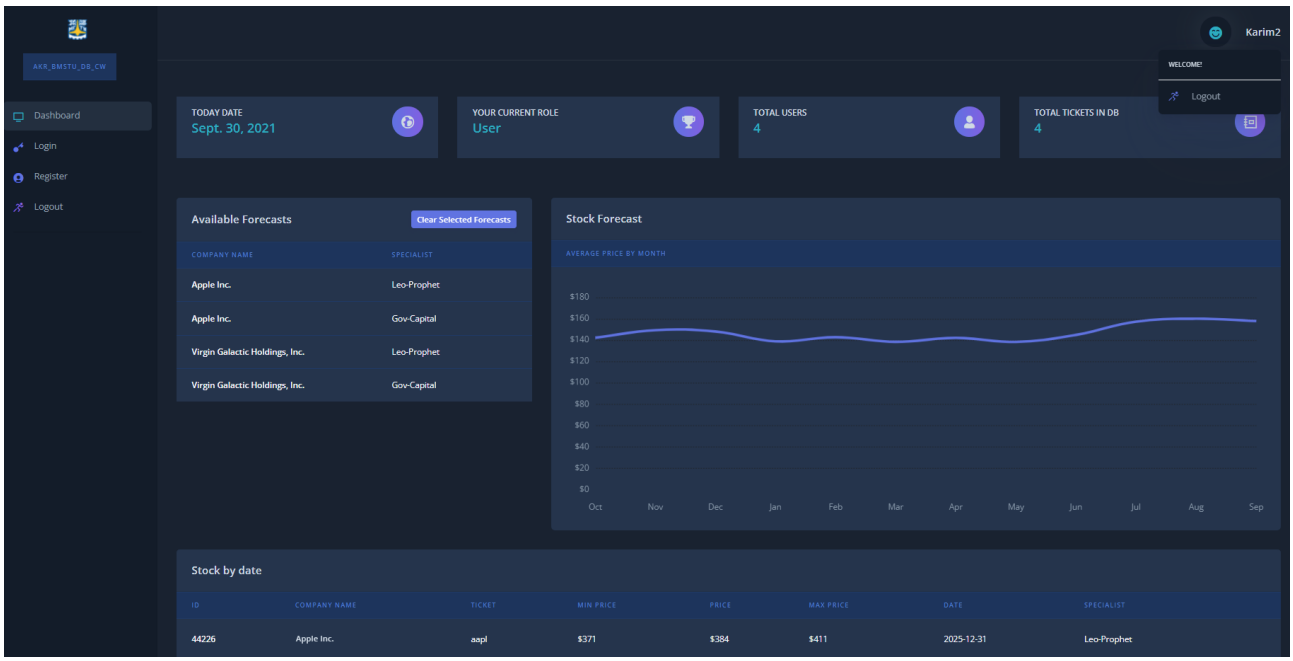

Рисунок 3.3.4 – главная страница сайта для обычного пользователя.

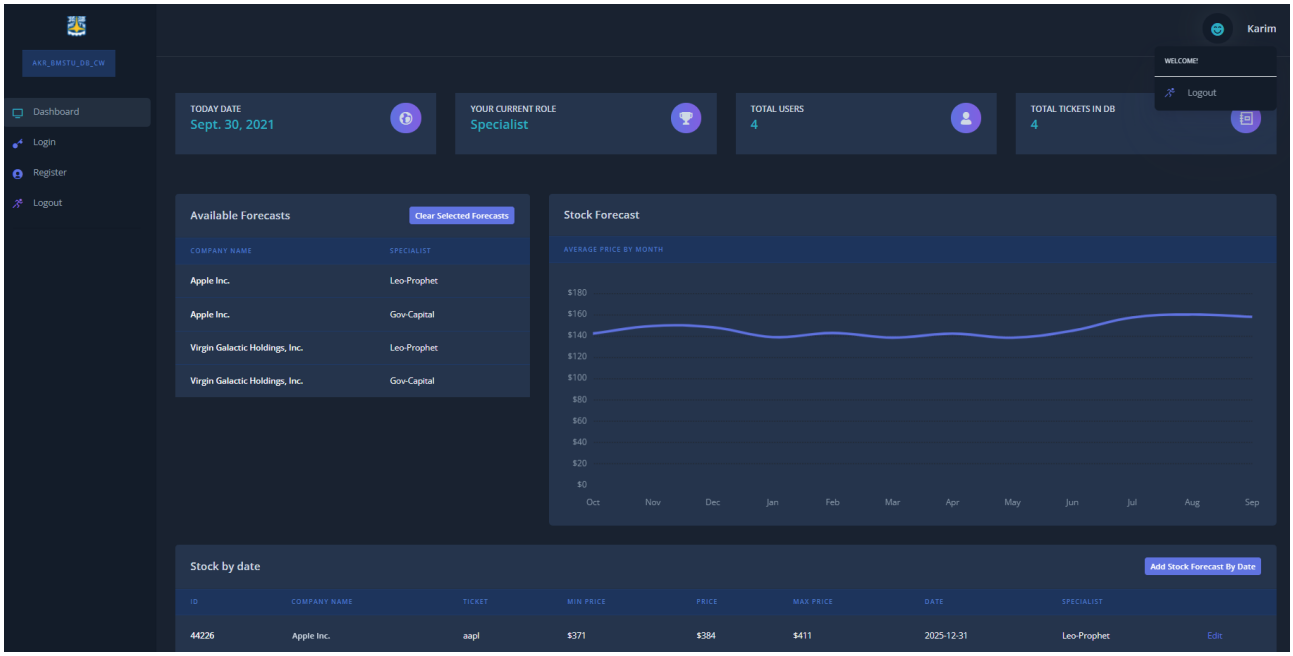

Рисунок 3.3.5 – главная страница сайта для специалиста.

Для удобства различные методы взаимодействия с базой данных с помощью интерфейса дополнительно защищены на уровне рендера страницы. У обычного пользователя нет соответствующих элементов страницы, к примеру кнопки “Add Stock Forecast By Date” или “Edit”, которые позволяют специалисту или администратору менять прогнозы внутри базы данных. У пользователей с уровнем доступа администратор в выпадающем меню при

нажатии на имя пользователя дополнительно есть доступ в секцию администратора Django.

Также предусмотрен выбор нескольких прогнозов, для составления усредненного графика выбранных прогнозов на рисунке 3.3.6.

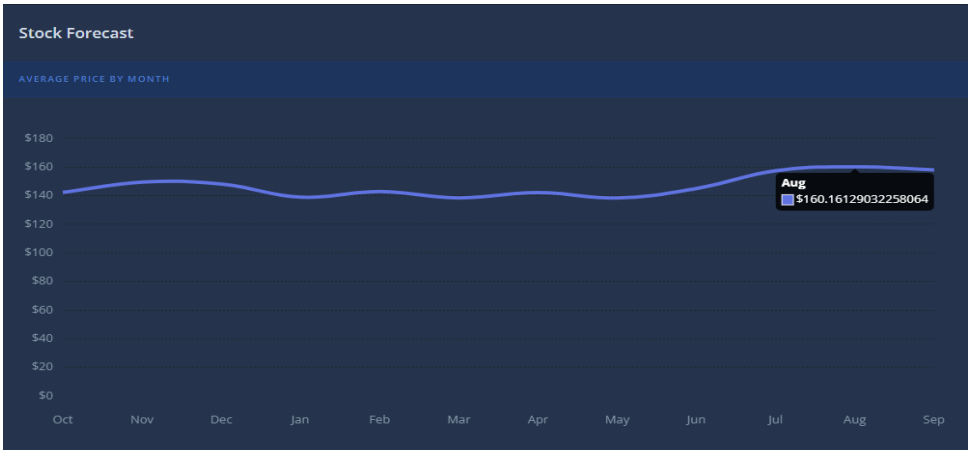

Рисунок 3.3.6 – Интерактивный график усредненного прогноза.

Для более подробного рассмотрения прогноза на конкретную дату внизу страницы предусмотрена таблица, отображенная на рисунке 3.3.7 и форма редактирования прогноза для пользователей с уровнем доступ администратор и специалист.

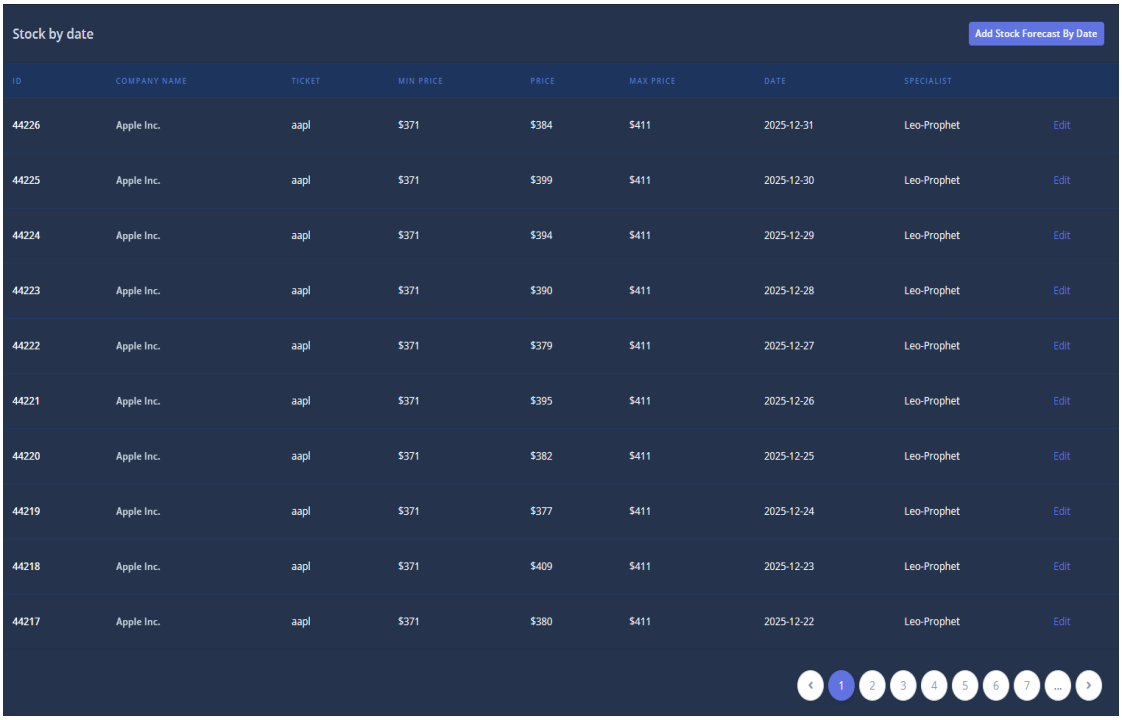

Рисунок 3.3.7 – таблица прогнозов на конкретную дату.

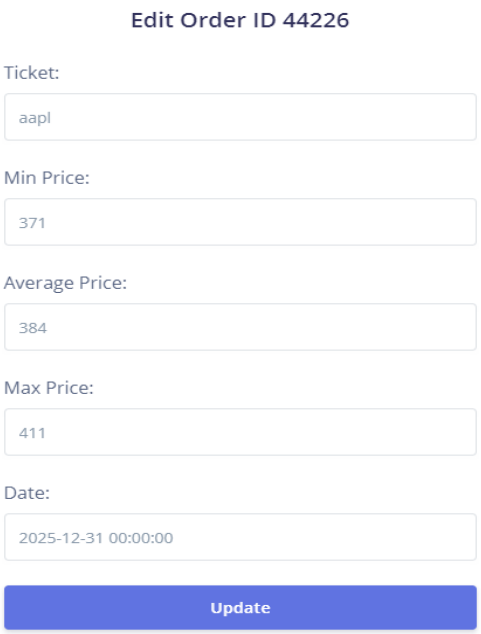

Рисунок 3.3.8 – форма редактирования прогноза на конкретную дату.

На рисунке 3.3.9 отображены доступные группы пользователей в панели администратора Django.

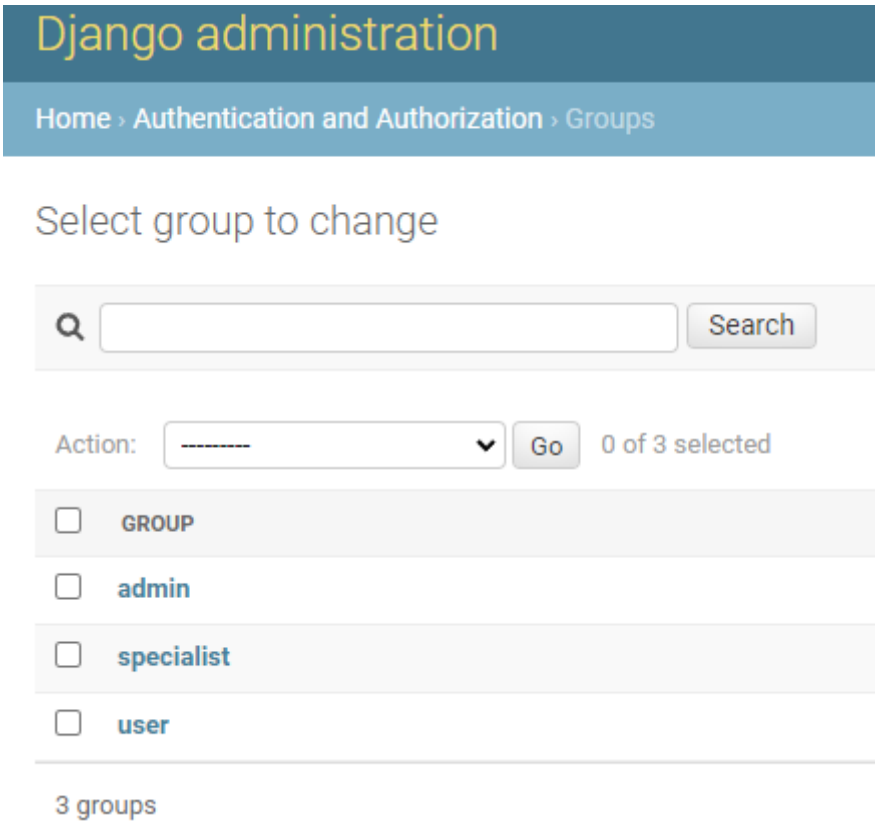

Рисунок 3.3.9 – доступные группы пользователей.

3.4 Выводы из технологической части

В данном разделе был выбран технологический стек разработки системы. Была приведена реализация компонентов доступа к базе данных и бизнес-логики, описан презентационный уровень приложения.

Заключение

Во время выполнения курсового проекта были рассмотрены существующие виды баз данных и СУБД. Было разработано программное обеспечение, которое предоставляет интерфейс к базе данных.

Программа реализована таким образом, чтобы давать возможность пользователям получать актуальную и правдоподобную информацию о прогнозах напрямую. Также пользователь может регистрироваться, осуществлять выбор интересующей информации. Специалист имеет возможность редактирования и добавление новой информации о прогнозах в существующую базу данных. Администратор имеет возможность полностью управлять базой данных при помощи панели администратора Django.

В ходе выполнения поставленной задачи были изучены возможности СУБД PostgreSQL, возможности разработки серверной части приложений на Python и языках Javascript, CSS и Html. Также были приобретены глубокие знания в области работы с фреймворком Django.

Список использованной литературы

[1] ISO/IEC TR 10032:2003 Information technology — Reference model of data management.

[2] PostgreSQL документация [Электронный ресурс]. –Режим доступа: https://postgrespro.ru/docs/postgresql/ (Дата обращения: 24.08.2021)

[3] Django documentation [Электронный ресурс]. -Режим доступа: https://docs.djangoproject.com/en/3.2/ (Дата обращения: 24.08.2021)

[4] Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: «Вильямс», 2006.