

МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЁТ ПО УЧЕБНОЙ ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКЕ

Студент
гр. БИН-21-01 _____ С. С. Колтунов

Руководитель,
доцент, канд. физ.-мат. наук _____ А. В. Тювеев

Владивосток 2024

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВВГУ»)

Институт информационных технологий и анализа данных

Кафедра информационных технологий и систем

Индивидуальное задание

на учебную технологическую (проектно-технологическую) практику

Студенту гр. БИН-21-1 Колтунову Сергею Сергеевичу

1 Характеристика предприятия (организации), являющегося объектом дальнейшей автоматизации (полное наименование, краткая информация, основные виды деятельности компании, миссия и основные бизнес-цели компании, номенклатура продукции или услуг, описание организационной структуры предприятия с описанием основных подразделений.)

2 Состояние и стратегия развития информационных технологий в организации (степень автоматизации процессов в подразделении компании, покрытие функциональных областей, ИТ-архитектура, определение уровня зрелости ИТ процессов по стандарту СММІ. Наличие в компании программно-аппаратных комплексов, технических устройств, корпоративных информационных систем и других ИС для эффективного управления предприятием.)

3 Обоснование выбора технологии проектирования на основе анализа существующих разработок



4 Техническое задание на разработку информационной системы по следующей структуре:

- 1) Общие сведения об информационной системе
- 2) Цели и задачи проекта автоматизации
- 3) Объект автоматизации
- 4) Требования к информационной системе
- 5) Этапы, сроки и результаты выполнения
- 6) Порядок контроля и приемки информационной системы
- 7) Требования к документированию

5 Срок сдачи отчета на кафедру: 12.07.2024

Руководитель,
Кандидат физико-математических наук

Задание получил:

Тюбеев А.В.

Колтунов С.С.

КАЛЕНДАРНЫЙ ПЛАН-ГРАФИК (ДНЕВНИК)
прохождения учебной технологической (проектно-технологической) практики сту-
дента «ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВВГУ»)

Студент Колтунов Сергей Сергеевич направляется для прохождения учебной технологической (проектно-технологической) практики в ФГБОУ ВО «ИТС», Институт креативных индустрий, г. Владивосток.

С 10 июня 2024 г. по 13 июля 2024 г.

№ п/п	Содержание выполняемых работ по программе	Сроки выполнения		Заключение и оценка руководителя или консультанта	Подпись руководителя или консультанта
		Начало	Окончание		
1	Сбор требований	10.06.24	17.06.24	хор	
2	Выбор инструментария	18.06.24	25.06.24	хор	
3	Выбор архитектуры	25.06.24	02.07.24	хор	
4	Разработка схем БД	01.07.24	09.07.24	хор	
5	Написание отчета	09.07.24	13.07.24	хор	

Согласовано:

Студент-практикант _____

С.С. Колтунов

подпись

дата

Руководитель от кафедры _____

А.В. Тюев

подпись

дата

Содержание

Введение.....	5
1 Описание и анализ деятельности ФБОУ ВО «ВВГУ».....	6
1.1 Общее описание ФБОУ ВО «ВВГУ».....	6
1.2 Организационная структура ФБОУ ВО «ВВГУ».....	6
1.3 Краткое описание института креативных индустрий ФБОУ ВО «ВВГУ».....	7
2 Ход работы.....	8
2.1 Согласование требований.....	8
2.2 Выбор инструментария.....	8
2.2.1 Базы данных.....	8
2.2.2 Backend.....	10
2.2.3 Frontend.....	10
3 Выбор архитектуры.....	12
4 Разработка схемы БД.....	15
Заключение.....	17

Введение

Данный отчет представляет результат практики, проведенной на предприятии (организации) с целью автоматизации процессов на этом предприятии. В ходе практики были выполнены задачи, направленные на проведение начального этапа разработки информационной системы предприятия.

Были поставлены следующие цели:

- Получение информации о процессе для автоматизации;
- Систематизация требований;
- Выбор необходимого инструментария, подходящего под задачу автоматизации;
- Получение необходимых знаний и навыков для реализации решения;

В первой части отчета представлена характеристика предприятия, которое является объектом автоматизации. В данном разделе предоставлена краткая информация о предприятии.

Следующая часть отчета посвящена описанию полученных требований, и формированию на их основе технического задания, выбору языка программирования(ЯП), а также выбору архитектуры и библиотек/фреймворков для их реализации.

Заключительная часть содержит в себе подведение итогов учебно-ознакомительной практики.

1 Описание и анализ деятельности ФБОУ ВО «ВВГУ»

1.1 Общее описание ФБОУ ВО «ВВГУ»

ФБОУ ВО «ВВГУ», основанный в 1967 году, является одним из ведущих высших учебных заведений на Дальнем Востоке России, специализирующимся на подготовке высококвалифицированных специалистов в области экономики, управления, сервиса, информационных технологий и гуманитарных наук. Университет сочетает в себе многолетние традиции и современные инновационные подходы в образовании и науке.

ФБОУ ВО «ВВГУ» является образовательным учреждением, выполняющим множество функций:

- Подготовка специалистов;
- Научные исследования;
- Социальная и культурная деятельность;
- Международная деятельность

1.2 Организационная структура ФБОУ ВО «ВВГУ»

На рисунке 1 предоставлена схема, описывающая основные элементы организационной структуры ФБОУ ВО «ВВГУ».



Рисунок 1 – Структура ФБОУ ВО «ВВГУ»

Основными составляющими этой структуры являются:

1. Ректорат

1.1. Ректор: главный руководитель университета, ответственный за стратегическое управление и представительство интересов университета на внешнем уровне

- 1.2. Проректоры: помощники ректора, которые курируют различные направления деятельности университета (например, учебно-методическую работу, научную деятельность, международные связи и т.д.).
- 1.3. Ученый совет: коллегиальный орган управления, решает ключевые вопросы развития университета, утверждает учебные программы и научные проекты
2. Учебные подразделения
 - 2.1. Институты: основные образовательные единицы, которые объединяют кафедры по близким направлениям подготовки.
 - 2.1.1. Директоры институтов: руководители институтов, ответственные за организацию учебного процесса и научной работы в институте
 - 2.2. Кафедры: структурные единицы факультетов, которые занимаются преподаванием и научной деятельностью по конкретным дисциплинам.
 - 2.2.1. Заведующие кафедрами: руководители кафедр, отвечающие за работу преподавателей и научных сотрудников на кафедре.
3. Административные подразделения:
 - 3.1. Отдел кадров: управляет вопросами, связанными с персоналом, включая прием на работу, аттестацию и повышение квалификации сотрудников.
 - 3.2. Финансовый отдел: отвечает за бюджетирование, финансовое планирование и учет.
 - 3.3. Студенческий офис: занимается консультированием и помощью студентам
4. Научные подразделения
 - 4.1. Лаборатории
 - 4.2. Научный отдел: занимаются проведением научных исследований и разработок в различных областях знаний.

1.3 Краткое описание института креативных индустрий ФБОУ ВО «ВВГУ»

Институт креативных индустрий Владивостокского государственного университета (ВВГУ) является важным подразделением университета, посвященным развитию и поддержке творческих профессий и индустрий. Он представляет собой центр образования, исследований и практической деятельности в области культуры, искусства и дизайна. Директором института является Ключко Инна Леонидовна. В институте две подчинённые кафедры: кафедра дизайна и технологий, колледж индустрии и моды.

2 Ход работы

2.1 Согласование требований

В результате диалога с руководителем от предприятия были получены сведения о проблеме, а также были получены базовые требования к решению. Проблемой является процесс получения внешних заказов, последующего поиска исполнителя в виде студента, а также полное отсутствие автоматизации организационных моментов, которые необходимы для получения и исполнения заказа. Эта проблема ведёт к потере времени и ресурсов директора Института креативных индустрий.

Предложенное решение состоит в разработке веб-приложения, которое позволяет автоматизировать этот процесс, позволяя заказчику общаться с исполнителями напрямую.

К предложенному решению со стороны руководителя были выставлены следующие требования:

- Наличие ролей заказчика/исполнителя
- Соответствие предоставленному дизайну
- Наличие функции чата

На основе анализа предложенного решения со стороны исполнителя были добавлены следующие требования:

- Наличие роли администратора
- Обеспечение высокого уровня защиты информации
- Обеспечение возможности масштабирования
- Адаптивность
- Высокая скорость работы
- Обеспечение отказоустойчивости

Таким образом, были согласованы основные требования к решению

2.2 Выбор инструментария

2.2.1 Базы данных

Для оптимизации работы с данными было принято решение об использовании нескольких баз данных. Потенциальные базы данных были поделены на группы в зависимости от цели их применения: работа с финансовыми данными, работа с данными веб-приложения (данными пользователей, данными заказов и т.д.), и данными для аналитики.

Для обработки и хранения финансовых данных важна скорость и доступность, поэтому для сравнительного анализа были выбраны NoSQL базы данных MongoDB, Cassandra и Redis.

MongoDB использует формат BSON (бинарный JSON) для хранения данных, что позволяет хранить данные без жёстких схем и предопределённых структур. Также MongoDB поддерживает различные типы индексов, то позволяет ускорить выполнение запросов. Тем не менее, MongoDB не поддерживает транзакции, а также в связи с отсутствием жёстких схем возникает сложность в обеспечении миграций и администрирования.

Cassandra, в отличие от MongoDB, имеет структуру. Тем не менее, она достаточно гибкая, благодаря поддержке динамических семейств колонок. Это позволяет эффективно моделировать различные типы данных в финансовых приложениях без необходимости предварительного определения схемы. К тому же, эта база данных поддерживает легкие транзакции и согласованность данных. Также, она основана на архитектуре с распределёнными узлами, что поддерживает доступность данных даже при сбоях отдельных узлов. Минусами Cassandra являются сравнительно большее потребление ресурсов из-за своей архитектуры, а также сложность администрирования и миграции.

Redis хранит данные в оперативной памяти, что обеспечивает высокую скорость доступа. В Redis используется простая модель ключ-значение, что делает его более простым для интеграции. Эта база данных поддерживает хэширование, хранение множеств и списков, что делает его полезным для разнообразного вида задач. Redis поддерживает транзакции, что позволяет обеспечить атомарность и консистентность операций. Минусы Redis: ограниченный объём данных по причине хранения их в оперативной памяти. Также, Redis недостаточно устойчив к отказам.

Таким образом, основываясь на требованиях, была выбрана база данных Redis из-за высокой скорости работы и обеспечения безопасности данных. Низкую отказоустойчивость можно обеспечить высокой частотой создания дампов.

Для работы с данными веб-приложения была выбрана база данных PostgreSQL за расширенные возможности SQL, высокую масштабируемость, производительность и гибкость, а также открытый исходный код и активную поддержку.

Для работы с данными для аналитики хорошо подходят NoSQL базы данных ClickHouse и Tarantool.

ClickHouse это колоночная база данных, что позволяет эффективно сжимать данные, экономя ресурсы. К тому же, ClickHouse изначально ориентирован на выполнение аналитических запросов и обработке большого объёма данных. Легкость вертикального и

горизонтального масштабирования делает ClickHouse мощным инструментом для работы с аналитическими данными

Tarantool предназначен для обработки больших объемов данных в режиме реального времени. Он обеспечивает высокую скорость выполнения операций чтения и записи благодаря оптимизированным структурам данных и архитектуре. К тому же у него есть встроенная поддержка Lua-скриптов, что позволяет интегрировать сложную логику и обращаться к внешним сервисам. Tarantool поддерживает разнообразные структуры данных, включая хэши, отсортированные множества, списки и графы. Это позволяет эффективно моделировать различные типы данных

Основываясь на этом, для работы с аналитическими данными была выбрана база данных ClickHouse за высокую скорость, экономию ресурсов и лёгкостью масштабирования.

2.2.2 Backend

Для разработки Backend-части приложения подходят множество языков. Для сравнения были выбраны языки Python, JavaScript, C#.

Python обладает огромным количеством библиотек и фреймворков для реализации backend-части веб-приложения. К тому же, благодаря своим компиляторам Python имеет достаточно высокую производительность.

JavaScript позволяет использовать единый язык как для написания frontend-части, так и backend-части, что упрощает синхронизацию. В дополнение к этому, JavaScript обладает наиболее удобными инструментами работы с JSON, который является стандартным форматом обмена данных в веб-приложениях. Тем не менее, JS не хватает производительности относительно других языков

C# является компилируемым языком программирования, что обеспечивает ему высокую производительность. К тому же, у него присутствует широкая поддержка интеграции с технологиями Microsoft. Минусы: C# тесно связан с платформой .NET, что может ограничить использование языка операционных системах, отличных от Windows

В итоге, для разработки был выбран язык Python. В качестве фреймворка был выбран FastApi, т.к. он является самой передовой, проработанной реализацией RestApi с активной поддержкой.

2.2.3 Frontend

Для разработки frontend-части приложения был выбран язык JS, как основной язык для разработки фронтенда. В качестве фреймворка был выбран Next.js. Основанный на фреймворке React, данный фреймворк обеспечивает наиболее мощный инструмент

разработки динамических веб-приложений, за счёт поддержки TypeScript, серверного и клиентского рендеринга, что позволяет ему выйти за пределы SPA, что улучшает его индексацию поисковыми алгоритмами, огромным количеством встроенных хуков и возможности пререндеринга. Встроенная интеграция Tailwind также позволяет настраивать внешнее представление данного фреймворка без необходимости прописывания огромных CSS файлов.

3 Выбор архитектуры

Выбор архитектуры зачастую играет решающую роль в создании веб-приложения. Правильная организация взаимодействия различных элементов приложения позволяет оптимизировать его работу, сохраняя ресурсы и время. Существует несколько вариантов построения веб приложения:

Монолитная архитектура: в данном варианте весь функционал сосредоточен в одном приложении, что обеспечивает простоту тестирования и развёртывания, а также высокую производительность. Несмотря на это, монолитные веб-приложения сложно масштабировать, так как приходится масштабировать приложение целиком, вместо отдельных частей. При продолжительном расширении монолитные приложения трудно поддерживать. Из-за высокой степени связанности компонентов, монолитные приложения обладают низкой отказоустойчивостью, т.к. неполадки в одном компоненте приводят к отказу всего приложения в целом.

Клиент-серверная архитектура: в данном варианте приложение делится на две основные части: фронтенд и бэкенд, взаимодействующие друг с другом через сетевые протоколы. По сравнению с монолитной архитектурой, обладает большей отказоустойчивостью, распределённой нагрузкой, повышенным уровнем безопасности. Тем не менее, клиент-серверная архитектура также обладает достаточно низкой отказоустойчивостью, а также трудностью масштабирования.

Микросервисная архитектура: в данном варианте приложение делится на разное количество сервисов, каждый из которых изолирован и отвечает только за свои задачи. Это позволяет достичь лёгкости масштабирования, а также хорошей отказоустойчивости, так как при выходе из строя одного сервиса другие могут продолжить свою работу из-за высокой степени изоляции. Тем не менее, микросервисная архитектура требует дублирования кода, хорошей проработки сетевой инфраструктуры, а также вызывает сложности в согласованности данных между базами данных.

Одностраничное приложение: в данном варианте сервис загружает одну страницу и динамически обновляет её при необходимости. Это убирает необходимость перезагрузки страниц, а также перехода между ними, что заметно улучшает UX. Также одностраничные приложения меньше нагружают сервер. Они просты для развёртывания и обеспечивают высокую скорость работы. Несмотря на это, одностраничные приложения могут вызывать трудности у поисковых систем, так как им сложно индексировать контент приложения, из-за того, что он не загружен. Также, данная архитектура работает с логикой на стороне клиента, что делает данные более уязвимыми

Проанализировав примеры приложений, написанных на основе данных архитектур, а также пересмотрев требования руководителя, была выбрана микросервисная архитектура, так как она обеспечивает наибольший уровень безопасности данных, отказоустойчивости системы, а меньшая скорость может быть скомпенсирована проработкой кода и инфраструктуры. Итоговая архитектура продемонстрирована на рисунке 2

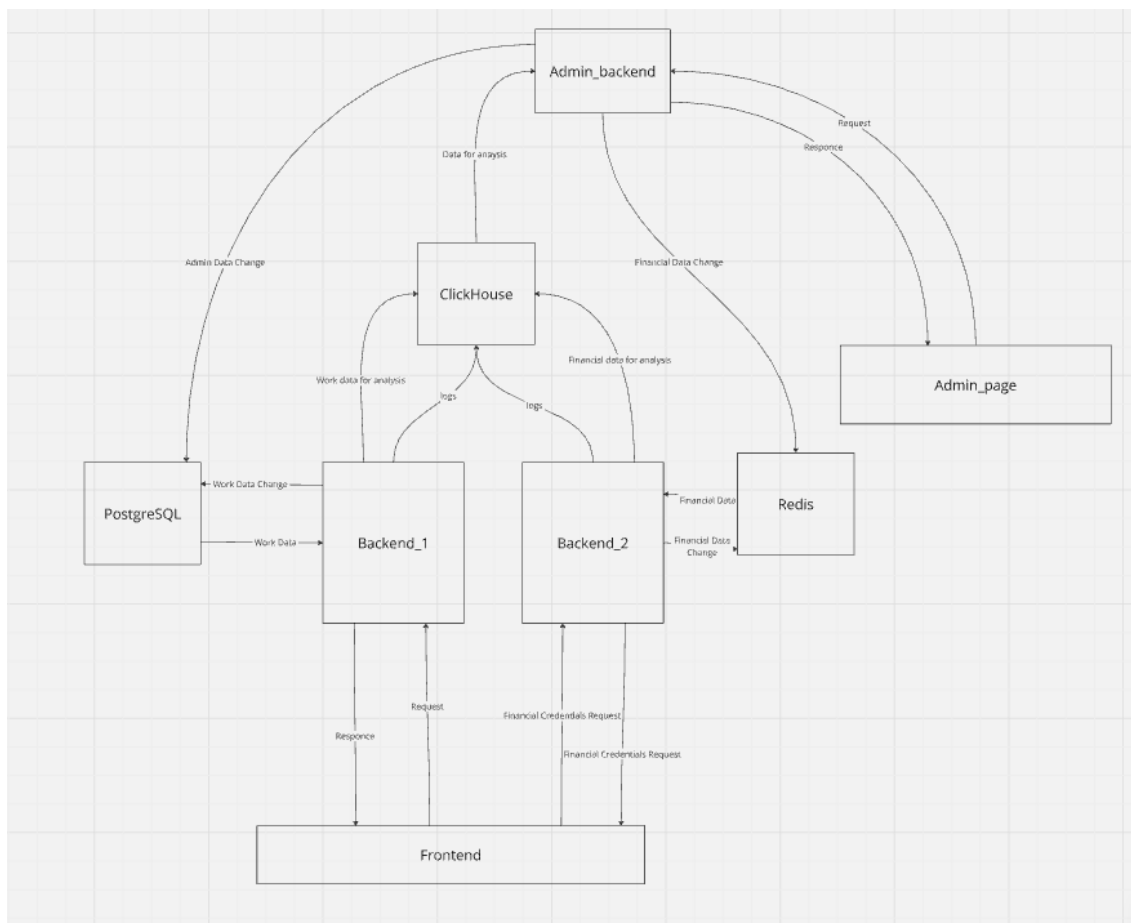


Рисунок 2 – Архитектура решения

Согласно данной схеме, для распределения нагрузки и обеспечения изоляции данных вместо монолитного backend-frontend приложения будет создано веб-приложение, в котором в зависимости от типа запроса будет выбираться нужный backend, после чего данный backend будет обрабатывать запрос.

При обработке данных приложения (**Work Data**), запрос типа **Work Request** с **Frontend** отправляется на **Backend_1**, который отвечает за работу с этими данными. У него есть права на запись и чтение в **PostgreSQL**, а также на запись логов и данных приложения в **ClickHouse**. Полученные данные обрабатываются внутри **Backend_1**, после чего ответ посылается на **Frontend**.

В случае работы с финансовыми данными (**Financial Data**), посылается запрос типа **Financial Data Request** на **Backend_2**, имеющий права на чтение и запись в **Redis**, а также на

запись логов и финансовых данных в ClickHouse. Полученные данные обрабатываются внутри Backend_2, после чего ответ посылается на Frontend.

Для взаимодействия с приложением пользователя с ролью администратор реализован отдельный frontend-backend сервис, что позволяет обеспечить дополнительную изоляцию, а следовательно защиту и разграничение уровней доступа к данным. Admin_page – фронтенд компонент данного сервиса, Admin_backend – бэкенд. Сервис будет использоваться для работы с данными для аналитики, но ему также выдан ограниченный доступ к записи и чтению в PostgreSQL и Redis для своевременного и удобного обеспечения поддержки и устранения ошибок.

Таким образом, с помощью микросервисной архитектуры в итоговом решении будет достигнут лучший уровень безопасности данных, отказоустойчивости системы, а также будет достигнута простота масштабирования.

4 Разработка схемы БД.

В соответствии с задачами БД должна быть разработана её схема для дальнейшей работы. В итоговых описаниях полей будут пропущены поля `created_at`, `updated_at`, т.к. они не отличаются друг от друга и обозначают время создания записи и её последней модификации. Поле «...» означает необязательные, не играющие функциональные роли поля которые по желанию могут быть добавлены.

При работе с финансовыми данными важно выделить две основных сущности: Счёт (Account) и транзакция (Transaction). Итоговая ERD Redis представлена на рисунке 3

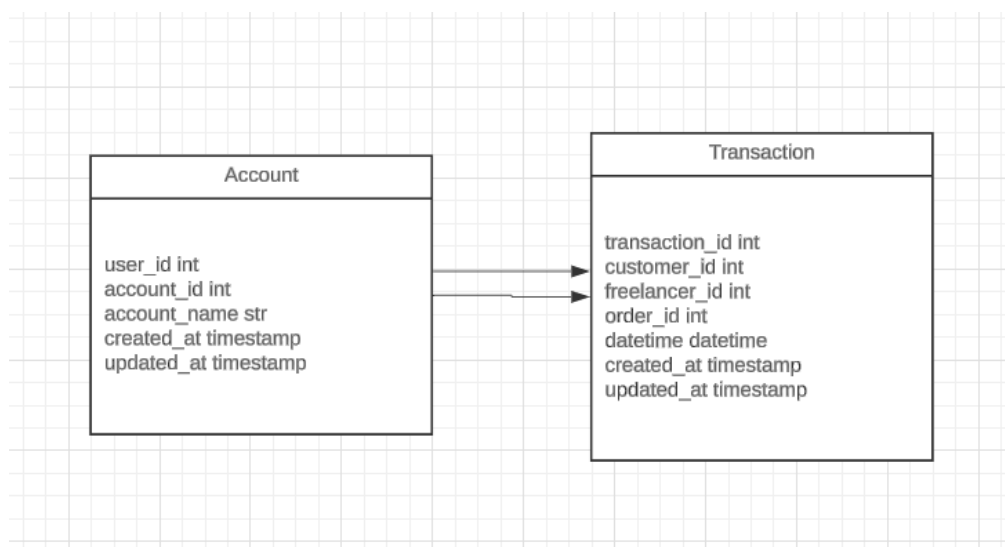


Рисунок 3 – Схема Redis

В таблице Account представлены следующие поля:

- `user_id`: идентификатор пользователя. Должен сходиться с идентификатором пользователя в других базах
- `account_id`: идентификатор счёта
- `account_name`: тип счёта

В таблице Transaction представлены следующие поля:

- `transaction_id`: идентификатор транзакции
- `customer_id`: идентификатор счёта заказчика
- `freelancer_id`: идентификатор счёта исполнителя
- `order_id`: идентификатор заказа. Должен сходиться с идентификатором заказа в других базах
- `datetime`: дата и время транзакции

База данных PostgreSQL должна содержать в себе данные для работы приложения, не включающие в себя финансовые данные. Здесь важно выделить следующие сущности: Пользователь (User), Заказ (Order). ERD для PostgreSQL представлен на рисунке 4

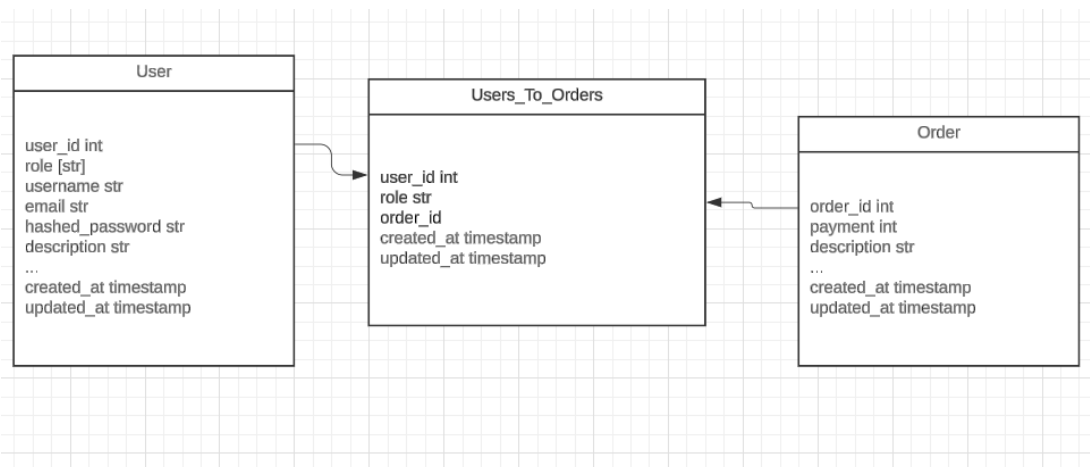


Рисунок 4 – Схема PostgreSQL

На данный момент невозможно предоставить схему базы данных ClickHouse, так как некоторые её поля, играющие ключевую роль в функционале, сильно зависят от содержимого логов бэкендов. Поэтому составление схемы ClickHouse будет произведено на последующих этапах построения системы.

Заключение

В ходе учебно-технологической практики были достигнуты все поставленные цели в той мере, в которой это было возможно на данном этапе. Были получены, дополнены и проанализированы требования к решению по автоматизации. На основе этого анализа был выполнен выбор инструментария и архитектуры, а также разработаны схемы БД. Результатом выполнения задач стало формирование готовой базы для дальнейшей реализации системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сведения об образовательной организации. Владивостокский государственный университет ВВГУ: [сайт]. – URL: <https://www.vvsu.ru/sveden/> (дата обращения 16.06.2024).
2. MongoDB: [сайт]. – URL: <https://www.mongodb.com> (дата обращения 22.06.2024).
3. Cassandra [сайт]. – URL: <https://cassandra.apache.org> (дата обращения 22.06.2024)
4. Разбираемся с Redis / Хабр: [сайт]. – URL: <https://habr.com/ru/companies/wunderfund/articles/685894/> (дата обращения 22.06.2024).
5. Архитектура веб-приложения: компоненты, слои и типы: [сайт]. – URL: <https://itanddigital.ru/webapplications> (дата обращения 29.07.2024)