МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Satbayev University
Институт информационных и информационных технологий Кафедра кибербезопасность, обработка и хранение информации

Мурзаев Алихан Рустамұлы

Разработка Интернет-Банкинга для Юридических Лиц

ДИПЛОМНАЯ РАБОТА

Специальность 5В070300 – Информационные системы

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Институт кибернетики и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой КБОиХИ
канд. техн. наук, доцент
_Н. А. Сейлова
«___»____20г.

ДИПЛОМНАЯ РАБОТА

На тему: Разработка Интернет-Банкинга для

Юридических Лиц

Специальность 5В070300 – Информационные системы

Выполнил**:** Мурзаев А. Р.

Научный руководитель
Магистр технических наук, лектор
_____Дуйсенбаева А. Н.

«28» мая2021г

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Институт кибернетики и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

Специальность 5В070300 – Информационные системы

УТВЕРЖДАЮ

Заведующий кафедрой КОиХИ

канд. техн. наук, доцент

 Н.А.Сейлова

“28” мая 2021 г.

Задание

на выполнение дипломной работы

Обучающемуся: Мурзаев Алихан Рустамұлы
Тема: Разработка Интернет-Банкинга для Юридических Лиц
Утверждена *приказом Ректора Университета №762-б от 24.11.2021 г.*
Срок сдачи законченной работы — *27.05.2021 г.*
Исходные данные к дипломному проекту: результаты преддипломной практики, результат обзора современного состояния по данной теме, сбор теоретического материала.
Краткое содержание дипломной работы:
а) Аналитический обзор современного рынка Автоматизированных Банковских Систем;
б) Архитектура ИС интернет-банкинга;
в) Разработка ИС интернет-банкинга;
г) Внедрение в эксплуатацию;
Рекомендуемая основная литература: *из 10 наименований*

ГРАФИК

подготовки дипломной работы

Наименования разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю**	**Примечание**
Аналитический обзор современного рынка Автоматизированных Банковских Систем	17.02.2021 г.	
Архитектура ИС интернет-банкинга;	11.04.2021 г.	
Разработка ИС интернет-банкинга	21.04.2021 г.	
Внедрение в эксплуатацию	16.05.2021 г.	

Подписи

консультантов и нормоконтролера на законченную дипломную работу с указанием относящихся к ним разделов работы

Наименование разделов	Консультанты, Ф.И.О. (уч. степень, звание)	Дата подписания	Подпись
Нормоконтроль	Кабдуллин М.А., магистр техн.наук, ассистент	19.05.2021	

Научный руководитель  Дуйсенбаева А. Н.

Задание принял к исполнению обучающийся  Мурзаев А. Р.

Дата “ 27 ” 01 2021 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Отзыв научного руководителя

Дипломная работа

Мурзаев Алихан Рустамұлы

Специальность 5В070300 – Информационные системы

Тема: Разработка Интернет-Банкинга для Юридических Лиц

Дипломная работа представляет собой выпускную квалификационную работу по специальности 5В070300 – «Информационные системы». Пояснительная записка состоит из введения, 4 глав, заключения, списка использованных источников и приложения. Автор дипломной работы поставленные задачи полностью выполнил и показал владение современными технологиями в предметной области. Дипломная работа выполнена на достаточном профессиональном уровне и содержит все необходимые сведения для такого рода работ. К замечаниям следует отнести незначительные стилистические ошибки. Считаю, что дипломная соответствует требованиям предъявляемым к выпускным квалификационным работам по специальности 5В070300 – «Информационные системы». Автор работы Мурзаев Алихан заслуживает присвоения академической степени бакалавра.

Научный руководитель

Магистр технических наук, лектор Дуйсенбаева А. Н.

“ 1 ” Июня 2021 г.

StrikePlagiarism.com

SATBAYEV UNIVERSITY

Report date 5/25/2021
Edit date ---

Metadata

Title
Разработка Интернет-Банкинга для Юридических Лиц

Author(s)
Мурзаев Алихан

Promoter
Асемгуль Дуйсенбаева

Organizational unit
ИКиИТ

List of possible text manipulation attempts

In this section, you can find information regarding text modifications that may aim at temper with the analysis results. Invisible to the person evaluating the content of the document on a printout or in a file, they influence the phrases compared during text analysis (by causing intended misspellings) to conceal borrowings as well as to falsify values in the Similarity Report. It should be assessed whether the modifications are intentional or not.

Characters from another alphabet	ß	0
Spreads	A→	0
Micro spaces	·	0
White characters	ß	0
Paraphrases (SmartMarks)	a	0

Record of similarities

Please note that high coefficient values do not automatically mean plagiarism. The report must be analyzed by an authorized person.

25
The phrase length for the SC 2

3820
Length in words

31975
Length in characters

Active lists of similarities

Scroll the list and analyze especially the fragments that exceed the SC 2 (marked in bold). Use the link "Mark fragment" and see if they are short phrases scattered in the document (coincidental similarities), numerous short phrases near each other (mosaic plagiarism) or extensive fragments without indicating the source (direct plagiarism).

The 10 longest fragments

Color of the text

NO	TITLE OR SOURCE URL (DATABASE)	NUMBER OF IDENTICAL WORDS (FRAGMENTS)

from RefBooks database (0.00 %)

NO	TITLE	NUMBER OF IDENTICAL WORDS (FRAGMENTS)

from the home database (0.00 %)

NO	TITLE	NUMBER OF IDENTICAL WORDS (FRAGMENTS)

from the Database Exchange Program (0.00 %)

NO	TITLE	NUMBER OF IDENTICAL WORDS (FRAGMENTS)

from the Internet (0.00 %)

NO	SOURCE URL	NUMBER OF IDENTICAL WORDS (FRAGMENTS)

List of accepted fragments (no accepted fragments)

NO	CONTENTS	NUMBER OF IDENTICAL WORDS (FRAGMENTS)

АННОТАЦИЯ

Данная дипломная работа посвящена разработке информационной системы интернет-банкинга, технологии дистанционного банковского обслуживания, удовлетворяющей современным требованиям рынка Казахстана. В настоящей работе разработана информационная система, предоставляющая доступ различным банковским продуктам через веб-браузер. Проведен анализ рынка автоматизированных банковских систем, составлены требования к ИС. Интернет-банкинг разработан с использованием языка программирования Golang, в качестве СУБД была использована Oracle RDBMS.

THE ANNOTATION

This thesis is devoted to the development of an information system for internet banking, a technology for remote banking services that meets the modern requirements of the Kazakhstan market. In this work, an information system has been developed which provides access to various banking products through a web browser. The analysis of the market of automated banking systems was carried out, the requirements for IS were drawn up. Internet banking was developed using the Golang programming language, Oracle RDBMS was used as a DBMS.

АҢДАТПА

Ұсынылған дипломдық жұмыс интернет-банкингтің ақпараттық жүйесін, сонымен қатар Қазақстан нарығының қазіргі заманға сай талаптарына жауап беретін қашықтықтан банктік қызмет көрсету технологиясын әзірлеуге бағытталған. Берілген дипломдық жұмыста веб-шолғыш арқылы түрлі банктік өнімдерге қол жеткізуді қамтамасыз ететін ақпараттық жүйе жасалынды. Автоматтандырылған банктік жүйелер нарығына талдау жүргізілді, ақпараттық жүйеге қатысты талаптар құрастырылды. Интернет-банкинг Golang бағдарламалау тілі арқылы әзірленді, дерекқор басқару жүйесі ретінде Oracle RDBMS қолданылды.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Применение информационных технологий и автоматизация бизнес-процессов оказывают сильное влияние на финансовую индустрию, так как количество денежных операций, которые совершают клиенты банков, значительно увеличивается с каждым годом. Перенос части обслуживания клиентов в режим дистанционного взаимодействия с использованием глобальной сети Интернет способен оптимизировать операционные расходы банковских организаций и значительно увеличить качество обслуживания.

Цель данной дипломной работы — исследование работы информационной системы дистанционного банковского обслуживания на примере разработки интернет-банкинга для юридических лиц.

Для достижения поставленной цели были установлены следующие задачи:

- исследовать рынок Автоматизированных Банковских Систем;
- изучить современные подходы к реализации систем Дистанционного Банковского Обслуживания;
- провести анализ используемых технологий для разработки веб-приложений;
- используя изученные материалы, спроектировать архитектуру информационной системы;
- реализовать основные модули системы Дистанционного Банковского Обслуживания;
- произвести анализ построенной системы и разработать план по ее улучшению.

Актуальность темы дипломной работы обусловлена растущим числом юридических лиц в Казахстане, пользующихся услугами банков второго уровня. Учитывая тот факт, что цифровизация сферы финансовых технологий позволяет обеспечивать клиентов более качественным опытом использования продукта, безопасность, стабильность работы, удобство использования системы играют ключевую роль в успешности продукта на рынке.

Практическая значимость дипломной работы состоит в разработке модулей интернет-банкинга, готовых к внедрению и полноценному использованию в современных реалиях Казахстанского рынка финансовых технологий.

1 АНАЛИТИЧЕСКИЙ ОБЗОР СОВРЕМЕННОГО РЫНКА АВТОМАТИЗИРОВАННЫХ БАНКОВСКИХ СИСТЕМ

1.1 Основной функционал типового интернет-банкинга в Казахстане

Банком является юридическое лицо, коммерческая организация, которая в состоянии законно осуществлять банковскую деятельность. Банковская представляет собой различные виды операций над банковскими продуктами, в том числе:

- депозиты
- различные счета
- аккредитивы
- гарантии

Интернет-банкинг представляет собой комплекс из аппаратного и программного обеспечения, основной целью которого является оказание банковских услуг в формате онлайн. При использовании интернет-банкинга от клиента банка не требуется посещение отделения или филиала для совершения операций над банковскими продуктами. Доступ в таком случае осуществляется посредством глобальной сети Интернет и любого устройства (смартфон, планшет, персональный компьютер), подключенного к ней.

Развитие сети Интернет и ее распространение во всех регионах Казахстана повлекло за собой внедрение так называемых Систем Дистанционного Банковского Обслуживания.

Клиентами банка могут являться:

- юридические лица
- физические лица

Дистанционное Банковское Обслуживание (ДБО) — это услуга удаленного доступа к банковским продуктам. Осуществляется подобный доступ посредством взаимодействия клиента банка с автоматизированными системами, либо с сотрудниками банка, клиентом которого он является через различные каналы коммуникации.

Изобретение и внедрение в эксплуатацию инновационных коммуникационных технологий влекло за собой появление различных видов дистанционного банковского обслуживания.

В таблице 1.1 приведены типы дистанционного банковского обслуживания, классифицированные по типам используемых аппаратных и программных средств коммуникации.

Таблица 1.1 — Типы дистанционного банковского обслуживания

Наименование	Краткое описание
Типичный “Банк-Клиент”	Предполагает использование заранее установленного клиентского программного обеспечения, которое будет хранить данные клиента банка (в том числе, выписки по различным типам счетов, документы по платежам). Канал связи в таком случае — телефонная связь или Интернет
Интернет-банкинг	Доступ к продуктам банка осуществляется через глобальную сеть Интернет и любое устройство, имеющее доступ к нему (мобильный телефон, персональный компьютер)
Мобильный банкинг	Чаще всего под мобильным банкингом понимают информационный сервис, позволяющий посредством телефонного звонка с оператором-сотрудником банка получить информацию по выпискам, остаткам и т.д. Однако, некоторые банки позволяют совершать денежные переводы и платежи
Внешние сервисы	Представляют собой специализированные системы: терминалы, банкоматы и т.д. Являются устройствами самостоятельного банковского обслуживания.

Важнейшим фактором при выборе системы ДБО является её безопасность. Это связано, в первую очередь, с относительно крупными операциями по денежным переводам между счетами; в том числе, между различными банками.

С точки зрения безопасности, наиболее надежным является интернет-банкинг. Причиной тому является использование устройства One-Time Password Token (OTP token), позволяющего определить пользователя с помощью фактора владения устройством. Данное устройство выдается клиенту банка при регистрации в отделении и позволяет генерировать одноразовые пароли для входа в систему.

Подобная защита зачастую дополняется СМС-паролем. Однако, использование только СМС-пароля не обеспечивает полную безопасность из-за существования различных уязвимостей мобильных сетей. В число таких уязвимостей входит возможность атаковать протокол SS7 (Signaling System 7), которая позволяет перехватывать СМС[1].

Следующим важным аспектом и характеристикой систем ДБО является удобство использования. Интернет-банкинг позволяет осуществлять операции с помощью любого устройства, имеющего доступ к сети Интернет с обеспечением необходимого уровня безопасности.

Учитывая все перечисленные факторы, была выбрана тема дипломной работы.

1.2 Анализ вендоров Автоматизированных Банковских Систем и поставляемого ими функционала

Автоматизированная Банковская Система (АБС) — это набор аппаратно-программных средств, объединенных сетью в защищенный контур, позволяющие производить автоматизированный контроль, учет и анализ основных банковских бизнес-процессов[2].

К обеспечивающим подсистемам АБС относятся:

- техническое оснащение
- информационное обеспечение
- системы коммуникации и каналы связи
- программное обеспечение
- системы безопасности

АБС строятся по модульному принципу, согласно которому предусматривается разделение системы на элементы согласно их функционалу. На рисунке 1.1 изображены основные строительные блоки Автоматизированной Банковской Системы, которые входят в состав любой реализации[3].

Рисунок 1.1 — Основные модули АБС

На рынке финансовых технологий в Казахстане преобладают следующие АБС:

- Compass Plus
- Colvir

Colvir — это АБС, разрабатываемая компанией Colvir Software Solutions. Система насчитывает более сотни функциональных модулей. В их число входят:

1. Главная книга бухгалтерского учета;
2. Счета и клиенты;
3. Расчетные и кассовые операции;
4. Зарплатные проекты;
5. Кассы;
6. Гарантии;
7. Аккредитивы;
8. Валютный контроль;
9. Налоговый инспектор;
10. Переводы;
11. Счета.

Компания представлена в более, чем 20 странах. На рисунке 1.2 изображены банки Казахстана[4], являющиеся пользователями продукции Colvir Software Solutions.

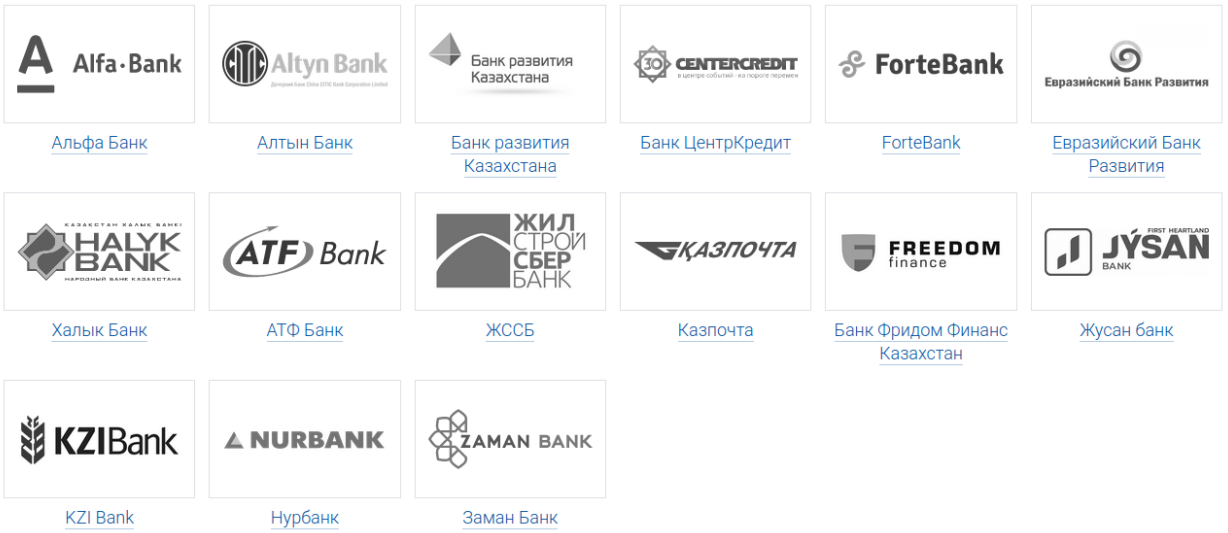

Рисунок 1.2 — Клиенты Colvir Software Solutions в Казахстане

В последние годы объемы корпоративных вкладов различных организаций продолжают расти (рисунок 1.3)[5]. В месячной динамике можно отметить 3 банка-лидера:

- ForteBank (плюс 72,9 млрд тг)
- Евразийский Банк
- Ситибанк

Данный факт свидетельствует о заинтересованности юридических лиц в использовании инновационных систем ДБО.

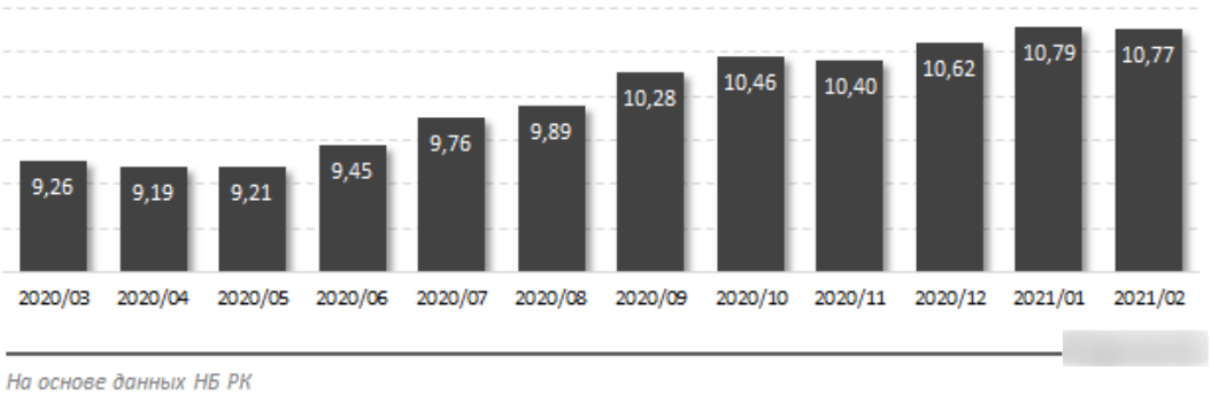

Рисунок 1.3 — Вклады физических лиц. Помесячная динамика (трлн тг)

2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ИС ИНТЕРНЕТ-БАНКИНГА

2.1 Требования к проектируемой системе и IT-инфраструктуре банковской организации

Проектируемое программное обеспечение “Интернет банкинг для юридических лиц” является прикладной системой, предназначенной для использования юридическими лицами, коммерческими организациями, индивидуальными предпринимателями, с целью получения различных услуг банка с помощью сети Интернет.

К основным областям применения относятся:

- Предоставление доступа к счетам клиента;
- Предоставление информации о передвижениях денег по клиентским счетам, платежах, переводах;
- Настройка списков бенефициаров платежей (получателей);
- Наблюдение за историей платежей и переводов;
- Создание, а также редактирование шаблонов платежей;
- Связь с сотрудниками банка посредством безопасного канала электронных сообщений;
- Осуществление продажи иностранных валют;
- Осуществление покупки иностранных валют.

Проектируемая система имеет несколько ролей пользователей:

1. Директор
2. Главный бухгалтер
3. Доверенное лицо
4. Менеджер

Требования к авторизации клиентов включают в себя список правил, изображенных в таблице 2.1.

Таблица 2.1 — Бизнес-правила системы

Код требования	Требования
Требование-1	Авторизация должна происходить согласно факторам аутентификации со следующими атрибутами: 1. Логин, пароль 2. Логин, пароль, OTP-token

	3. Логин, пароль, SMS-OTP
Требование-2	Время истечения одноразовых паролей — 120 секунд
Требование-3	После 3 неудачных попыток аутентификации учетная запись пользователя должна быть заблокирована
Требование-4	Доступ пользователей к банковским продуктам возможен при наличии действующего Договора и прохождении регистрации в Системе
Требование-5	Каждому пользователю системы присваивается определенный уровень доступа, который ограничивает определенные действия
Требование-6	Доступ клиента к интернет-банкингу должен осуществляться через WEB-приложение с использованием Браузера
Требование-7	Канал связи (протокол соединения) должен быть защищен шифрованием (HTTPS)
Требование-7	Осуществление переводов и платежей через систему интернет-банкинг должно происходить посредством создания электронных документов, которые формируются в системе автоматически и соответствуют действующему законодательству Республики Казахстан
Требование-8	Клиент осуществляет оплату электронных банковских услуг согласно тарифам, установленным Банком, действующим в момент пользования услуг
Требование-9	Система должна предусматривать прикрепление менеджера к пользователю
Требование-10	Менеджер должен иметь возможность вести переписку с пользователем посредством защищенного канала связи электронными сообщениями в рамках модуля электронной почты и онлайн-чата

Авторизация пользователей в системе должна происходить по схеме, представленной на рисунке 2.1.

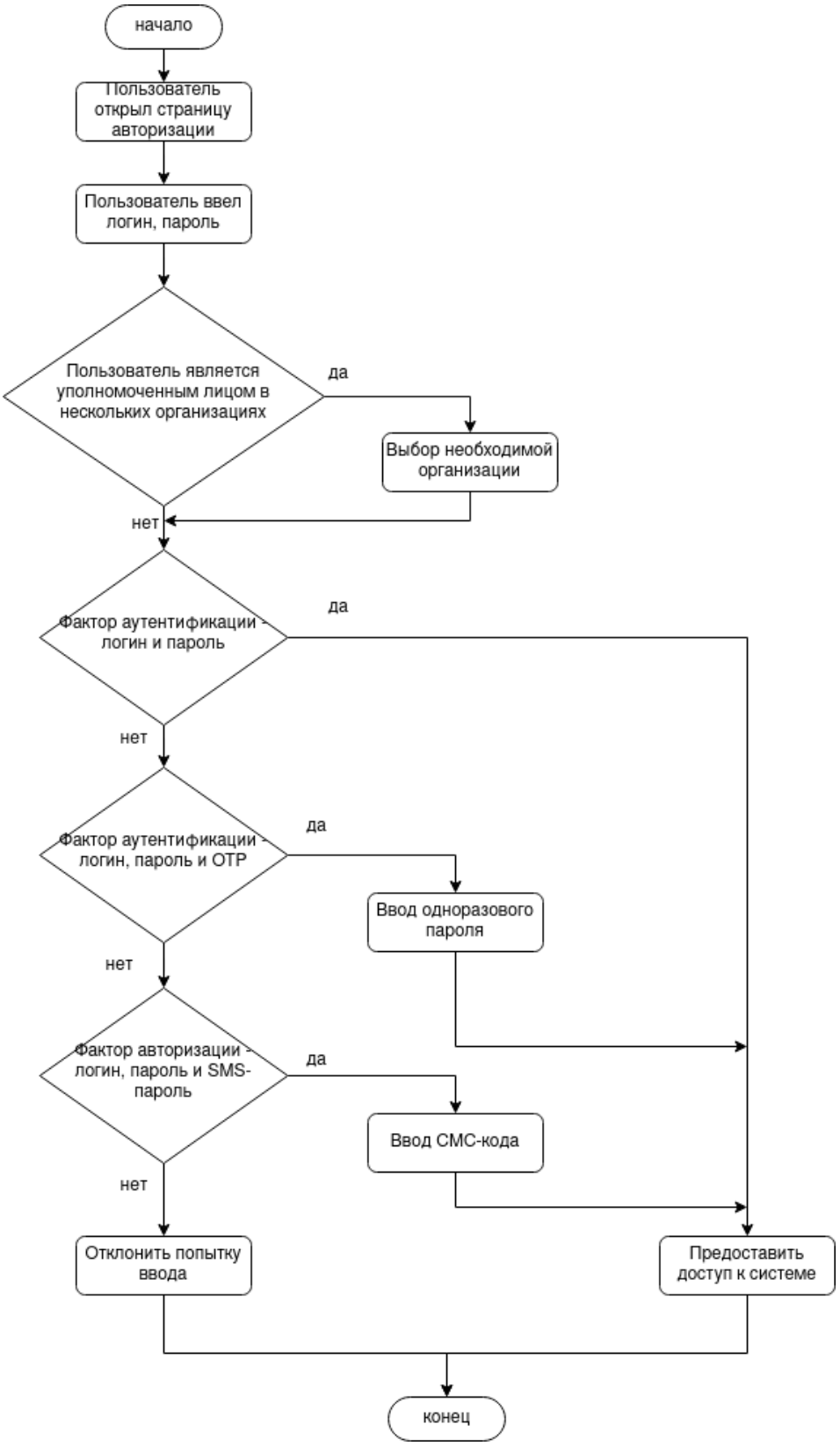

Рисунок 2.1 - Схема авторизации пользователей в системе

Основным функционалом системы является перевод денег в национальной валюте. На рисунке 2.2 представлена диаграмма переводов в тенге.

Рисунок 2.2 — Диаграмма переводов в тенге

Для развертывания информационной должны быть удовлетворены соответствующие требования к IT-инфраструктуре банковской организации. К серверу баз данных и подключенным каналам связи предъявляются аппаратные требования, описанные в официальной документации “ORACLE SPARC T8-2 Server”[12].

2.2 Анализ и сравнение современных способов реализации веб-приложений

При проектировании и разработке информационной системы необходимо учитывать множество требований, которым она должна удовлетворять. В их число входит доступность системы, консистентность данных, скорость работы, количество активных пользователей.

Архитектура приложения — это совокупность решений, касающихся организации программной системы. Кроме того, архитектура ПО:

- Формализует выбор составляющих структурных элементов и интерфейсов их взаимодействия;
- Описывает принципы взаимодействия выбранных элементов.

Для реализации системы ДБО была выбрана клиент-серверная архитектура. Подобная схема подразумевает существование двух систем:

1. Клиентское приложение (предоставляет пользовательский интерфейс, выполняет отправку запросов к серверному приложению).
2. Серверное приложение (предоставляет интерфейс к слою данных приложения, выполняет запросы клиентского приложения, выполняет различные фоновые задачи).

Данные составляющие взаимодействую друг с другом посредством сети Интернет (рисунок 2.3)

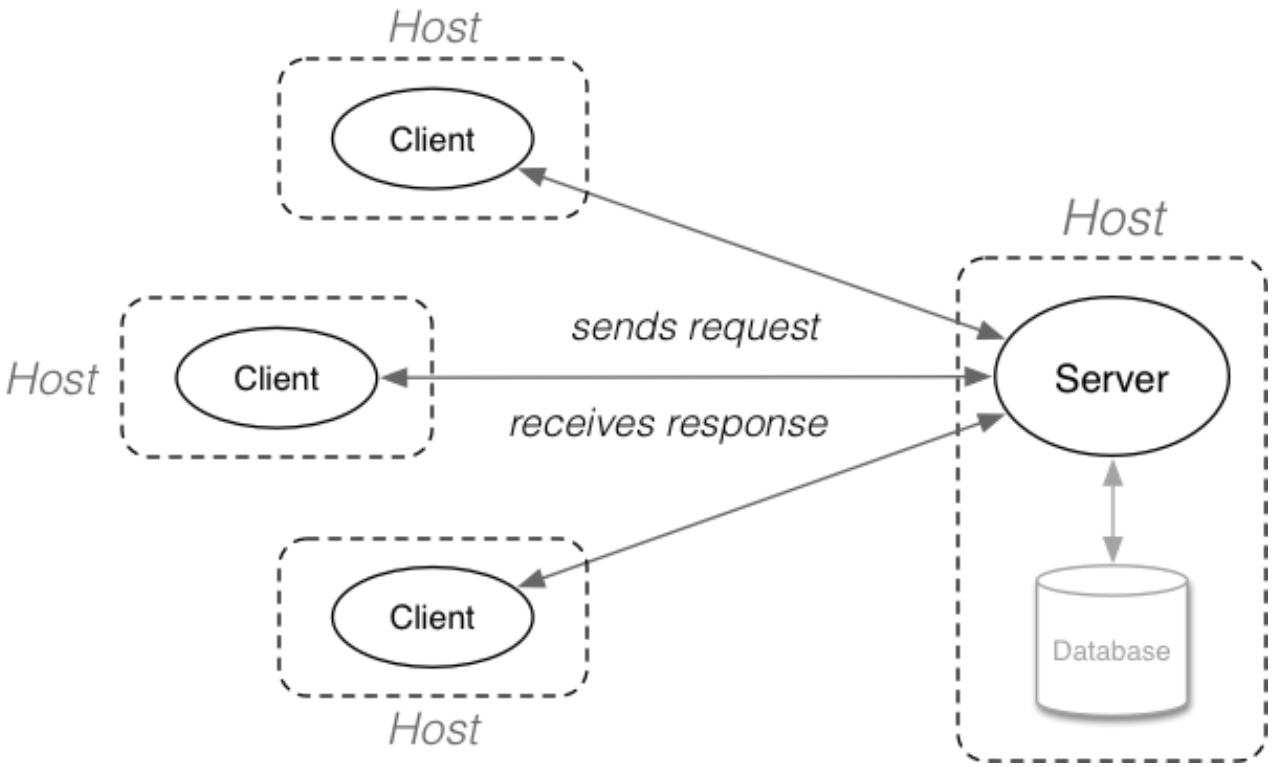

Рисунок 2.3 — Схема взаимодействия клиентов и сервера

При построении информационной системы с перечисленными элементами, серверная часть приложения должна обслуживать запросы от различных клиентов (браузерные веб-приложения, мобильные приложения, настольные приложения), предоставлять доступ к API (Application Programming Interface) с использованием различных протоколов передачи данных (HTTP, RPC), выполнять фоновые задачи, обмениваться данными с внешними сервисами.

Для разработки серверной части информационной системы применяется два подхода к организации кодовой базы:

1. Монолитная архитектура;
2. Микросервисная архитектура.

Программа, разработанная в виде монолита, представляет собой приложение, имеющее единую точку входа, и доставляется через единое развертывание. Из этого следует, что вся бизнес-логика сконцентрирована в одном репозитории, а внутренние элементы, зачастую, обладают высокой связностью. По мере роста приложения, разрастается кодовая база, хранящаяся, зачастую, в едином репозитории. На рисунке 2.4 изображена схема монолита

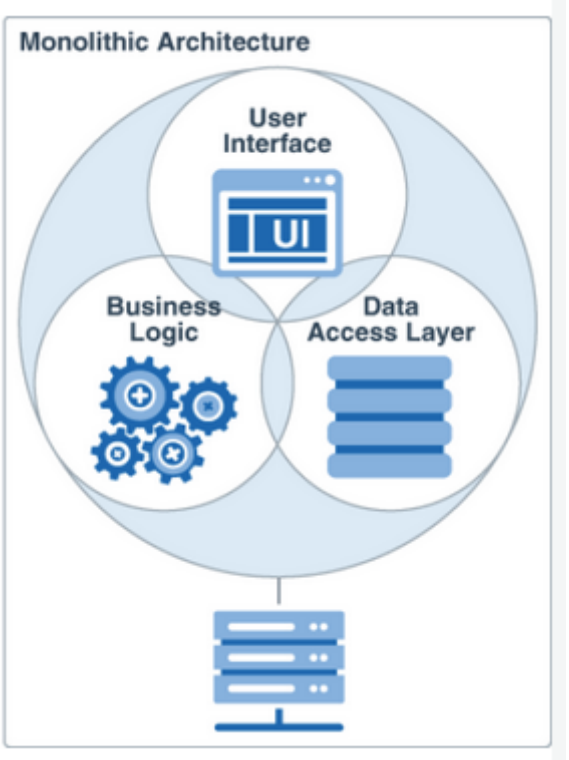

Рисунок 2.4 — Схема монолитного приложения

В таблице 2.2 отражены основные преимущества и недостатки монолитного подхода к разработке программных продуктов[6].

Таблица 2.2 — Анализ монолитного подхода

Преимущества	Недостатки
Легкая реализация	При кратном увеличении кодовой базы составляющие элементы монолита приобретают высокую связность
Несложная реализация сквозных тестов	Замедляется процесс разработки при большой кодовой базе
Легкое развертывание	Масштабирование и репликация

	усложняется при разрастании приложения

В противовес монолитной архитектуре, многими компаниями используется микросервисная (рисунок 2.5). При таком подходе к организации компонентов приложения, кодовая база разделена на отдельные модули с ограниченной зоной ответственности в рамках бизнес-логики.

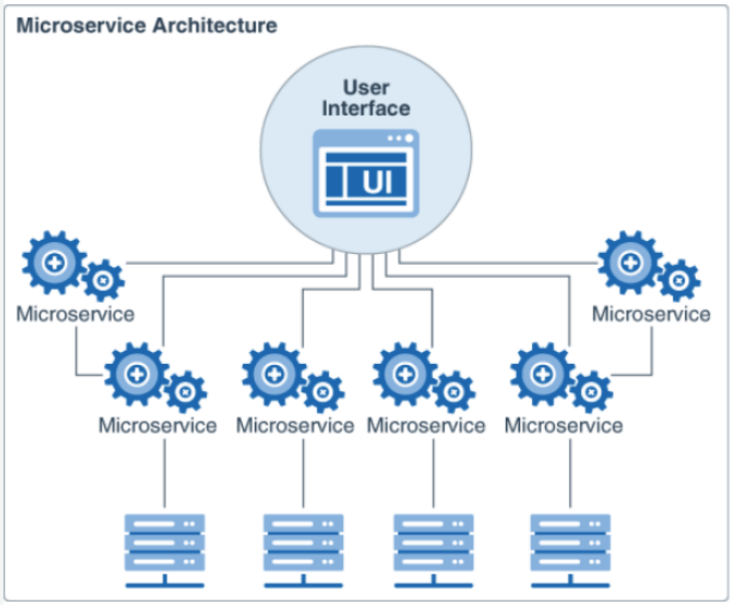

Рисунок 2.5 — Микросервисная архитектура

В таблице 2.3 отражены основные преимущества и недостатки микросервисной архитектуры[7].

Таблица 2.3 — Анализ микросервисного подхода

Преимущества	Недостатки
Разграничение ответственности между различными командами разработки (за разработку и поддержку каждого микросервиса может отвечать отдельный разработчик или целая команда)	Требования к компетенциям разработчиков гораздо выше
Из-за небольшого размера кодовой базы разработчикам легче вносить изменения в бизнес-логику	Требуется усложненная IT-инфраструктура

микросервиса	
Развертывание небольших по размеру микросервисов обходится дешевле по сравнению развертыванием монолита в рамках непрерывной доставки	Усложняется межсервисная коммуникация, необходима реализация распределенных транзакций для обеспечения консистентности данных

2.3 Сервисная архитектура

После проведения сравнительного анализа между двумя подходами к реализации серверной части приложения, была выбрана сервисная архитектура, которая представляет собой гибрид монолитной и микросервисной архитектур. Данный метод позволяет воспользоваться преимуществами и нивелировать недостатки двух рассмотренных подходов.

Спроектированная архитектура изображена на рисунке 2.6.

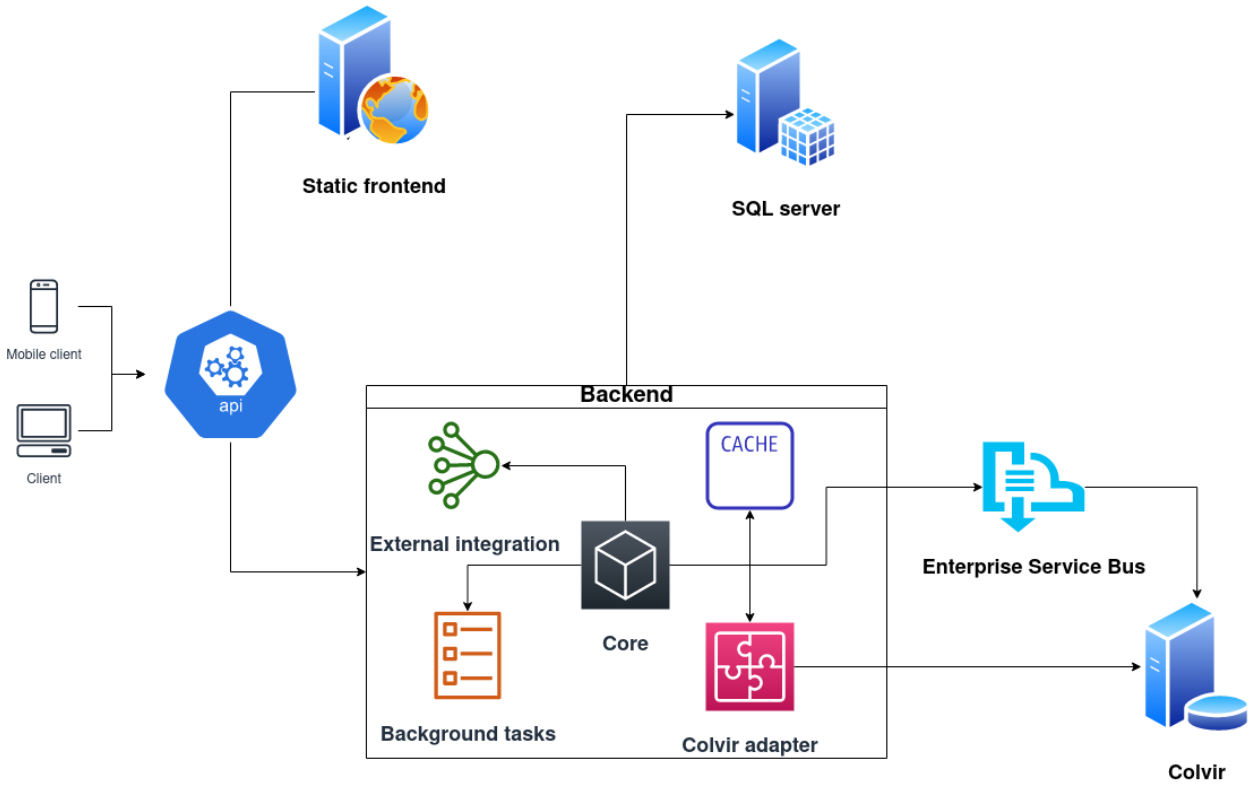

Рисунок 2.6 — Архитектура системы ДБО

В рамках проектирования архитектуры информационной системы были выделены следующие сервисы:

1. Core. Отвечает за основной функционал интернет-банкинга: создание и обработка платежей, взаимодействие со слоем данных.
2. Background tasks. Отвечает за выполнение запланированных фоновых задач. Например, обновление статусов платежей.
3. Cache. Отвечает за кэширование часто запрашиваемых данных. Сервис был реализован с использованием Redis (СУБД класса NoSQL).
4. Colvir adapter. Является заменой ESB (Enterprise Service Bus).

2.4 Описание информационного обеспечения

Неотъемлемой частью любой информационной системы является совокупность информационных ресурсов — информационное обеспечение (ИО). Слой информационного обеспечения характеризует состояние объектов, управление которыми происходит в информационной системе, и необходимо для поддержания высокого качества управления предприятием за счет следующих свойств ACID[9]:

- Атомарность;
- Согласованность;
- Изолированность;
- Прочность.

В таблице 2.4 приведен список таблиц, которые активно используются в приложении, и которые были построены на этапе создания физической модели базы данных. В таблицах 2.5-2.12 показана физическая структура базы данных, которая была разработана в рамках реализации проекта.

Таблица 2.4 — Перечень таблиц

Название	Краткое описание
DOCUMENT_STATE	состояние документа (платежа)
DOCUMENT_TYPE	содержит данные о типах документов
CURRENCY	содержит данные о валютах
CUSTOMER	хранит данные организаций-клиентов
ACCOUNTS	хранит данные о счетах клиентов

DOCUMENT	обобщенная таблица с данными о документах
PAYMENT	обобщенная таблица с данными о платежах
DOMESTIC_TRANSFER	содержит данные о переводах в национальной валюте

Таблица 2.5 — Структура таблицы “ACCOUNTS”

Наименование поля	Тип данных
ID	NUMBER(18)
CUSTOMER_ID	NUMBER(18)
ACCOUNT_NUMBER	VARCHAR2(34)
ACCOUNT_TYPE	VARCHAR2(32)
CURRENCY	VARCHAR2(3)
OPENED	DATE(7)
CLOSED	DATE(7)
BALANCE	NUMBER(22)
STATUS	VARCHAR2(32)

Таблица 2.6 — Структура таблицы “CURRENCY”

Наименование поля	Тип данных
ISO_CODE	VARCHAR2(3)
NAME	VARCHAR2(255)
DIGITAL_CODE	VARCHAR2(3)

Таблица 2.7 — Структура таблицы “CUSTOMER”

Наименование поля	Тип данных
ID	NUMBER(18)
NAME	VARCHAR2(120)
TAX_CODE	VARCHAR2(12)
DISTRICT	VARCHAR2(200)

ZIP	VARCHAR2(20)
CITY	VARCHAR2(200)
ADDRESS	VARCHAR2(500)
REGISTERED	DATE(7)
CONTRACT_STATUS	VARCHAR2(10)
CREATED	DATE(7)
DOC_REG_PLACE	VARCHAR2(500)
ORG_TYPE	VARCHAR2(100)
OWNERSHIP_CODE	VARCHAR2(100)

Таблица 2.8 — Структура таблицы “DOCUMENT”

Наименование поля	Тип данных
ID	NUMBER(18)
DOCUMENT_TYPE	VARCHAR2(30)
DOCUMENT_STATE	VARCHAR2(30)
CUSTOMER_ID	NUMBER(18)
CREATED	TIMESTAMP(6)(11)
UPDATED	TIMESTAMP(6)(11)
DOCUMENT_NUMBER	VARCHAR2(100)
BANK_RESPONSE	VARCHAR2(2000)
INFO	VARCHAR2(420)

Таблица 2.9 — Структура таблицы “DOCUMENT_STATE”

Наименование поля	Тип данных
DOCUMENT_STATE	VARCHAR2(30)
NAME	VARCHAR2(250)

Таблица 2.10 — Структура таблицы “DOCUMENT_TYPE”

Наименование поля	Тип данных
DOCUMENT_TYPE	VARCHAR2(30)
IS_DELETED	NUMBER(1)

Таблица 2.11 — Структура таблицы “DOMESTIC_TRANSFER”

Наименование поля	Тип данных
ID	NUMBER(18)
BUDGET_CLASSIFICATION_CODE	VARCHAR2(32)
PAY_DATE	DATE(7)

Таблица 2.12 — Структура таблицы “PAYMENT”

Наименование поля	Тип данных
ID	NUMBER(18)
ACCOUNT_ID	NUMBER(18)
AMOUNT	NUMBER(22)
VALUE_DATE	DATE(7)
PURPOSE	VARCHAR2(4000)
BENEFICIARY_NAME	VARCHAR2(1000)
BENEFICIARY_TAX_CODE	VARCHAR2(140)
BENEFICIARY_ACCOUNT	VARCHAR2(34)
COMMISSION	VARCHAR2(500)

Целостность данных в системе обеспечивается использования реляционной модели, которая, в свою очередь, связывает таблицы с помощью первичных и вторичных ключей[9].

2.5 Программное обеспечение

Программное обеспечение (ПО) является неотъемлемой частью информационной системы и представляет собой совокупность различного класса программ. В структуру программного обеспечения были включены:

1. Системное. В рамках разработки интернет-банкинга для юридических лиц была использована операционная система Ubuntu 20.04.
2. Инструментальное. Для написания серверной части приложения был использован язык программирования Golang версии 1.16. Данный язык программирования обладает следующим рядом

преимуществ, которые, в конечном итоге, повлияли на конечное решение:

a. конкурентная модель языка;
b. параллелизм;
c. растущая популярность языка;
d. достаточное количество кадров на рынке труда Казахстана с соответствующими компетенциями;
e. высокая производительность;
f. высокая скорость сборки проекта с большой кодовой базой;
g. простота синтаксиса.

3. Прикладное. В качестве прикладного ПО были использованы интегрированная среда разработки JetBrains Goland, Self-hosted Gitlab, git, Kubernetes, Docker, СУБД Oracle Enterprise Edition.

Структура программного обеспечения изображена на рисунке 2.7.

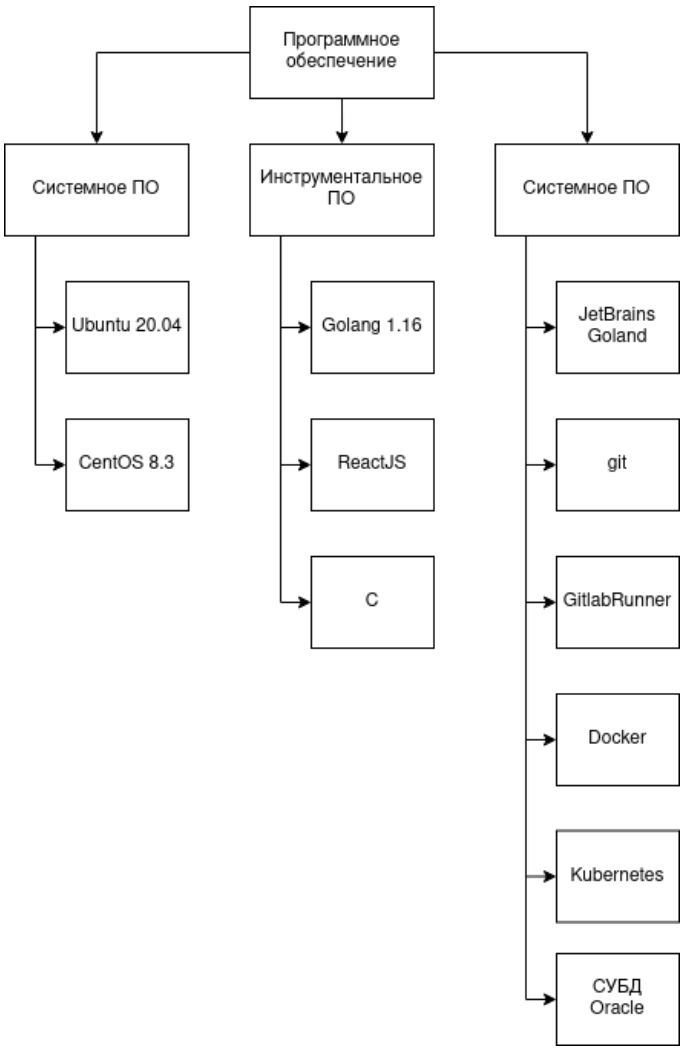

Рисунок 2.7 — Структура программного обеспечения

3 РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ ИНТЕРНЕТ-БАНКИНГА

3.1 Разработка основных модулей

Для разработки информационной системы было составлено техническое задание. Для учета затраченного времени на разработку была использована коммерческая система для отслеживания ошибок Jira, разработанная в компании Atlassian.

На рисунке 3.1 изображены банковские продукты, над которыми производятся различные операции в системе.

Рисунок 3.1 — Схема банковских продуктов

Наиболее часто используемым модулем системы ДБО является модуль переводов в национальной валюте. Для совершения каких-либо операций над переводами в тенге (создание, редактирование, удаление, подписание), пользователь должен обладать соответствующими привилегиями в системе и иметь доступ к счетами организации, уполномоченным лицом которой он является. Для взаимодействия с данным модулем необходимо перейти по ссылке “Переводы в валюте” в списке доступных операций (рисунок 3.2).

Для организации системы контроля доступа к продуктам была использована библиотека Casbin, которая поддерживаем следующие модели контроля доступа:

- ACL (Access Control List);
- BRAC (Role Based Access Control);
- ABAC (Attribute Based Access Control).

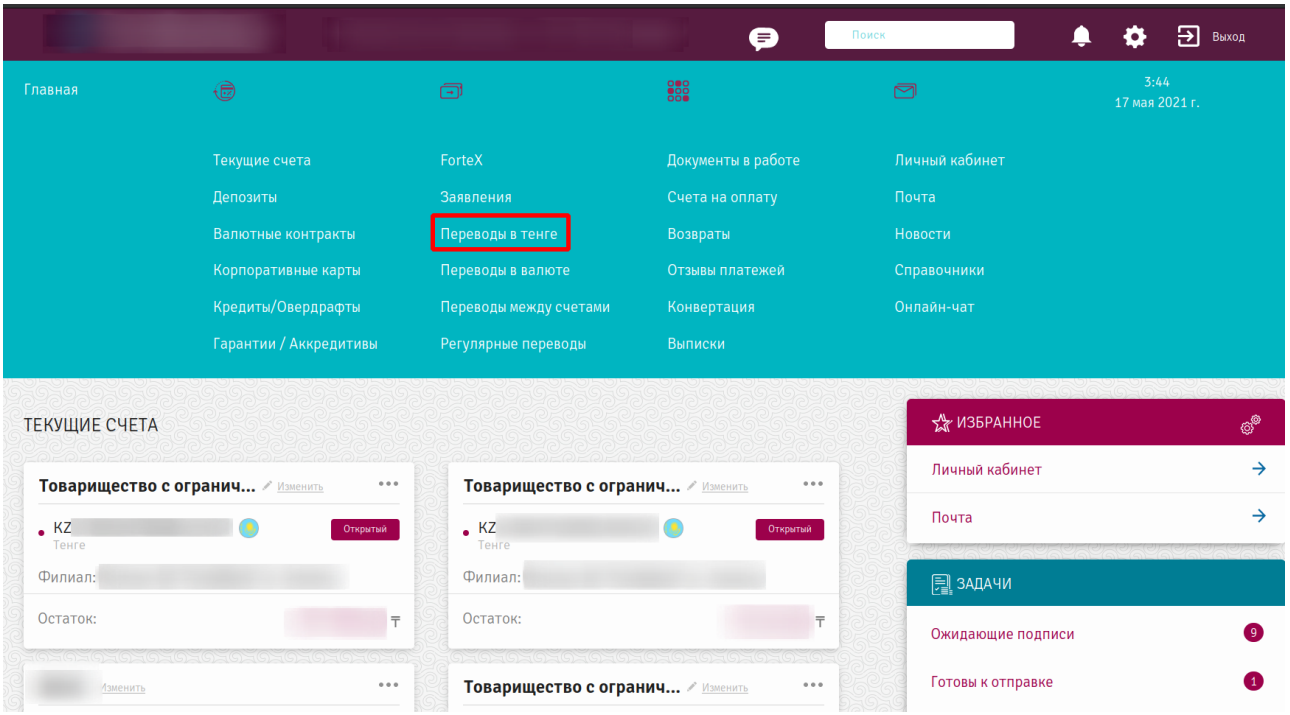

Рисунок 3.2 — Список продуктов на главном экране

Для просмотра списка платежных документов, а также актуальных статусов платежей был разработан фильтр, который позволяет выполнять поиск по датам, статусам и типам документов (рисунок 3.3).

Создание документа происходит в форме создания платежа (рисунок 3.4). Для предоставления более качественного опыта использования приложения, а также для увеличения скорости работы уполномоченных лиц компаний некоторые поля заполняются автоматически актуальными данными о текущей компании (отправитель, банк отправителя). Предусмотрена возможность создания так называемых “шаблонов” документов, которые позволяют снизить необходимость ручного ввода данных к минимуму при создании однотипных платежей. Кроме того, был реализован поиск банков по БИК (банковский идентификационный код) во время редактирования платежного поручения. Список доступных операций над вновь созданным документам представлен на рисунке 3.5.

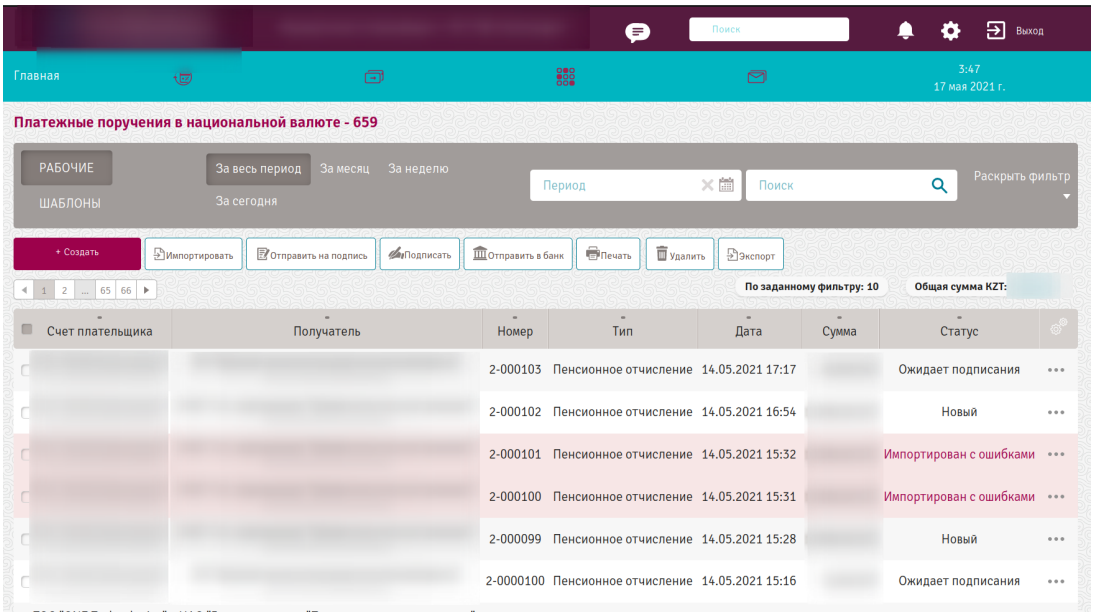

Рисунок 3.3 — фильтр по документам

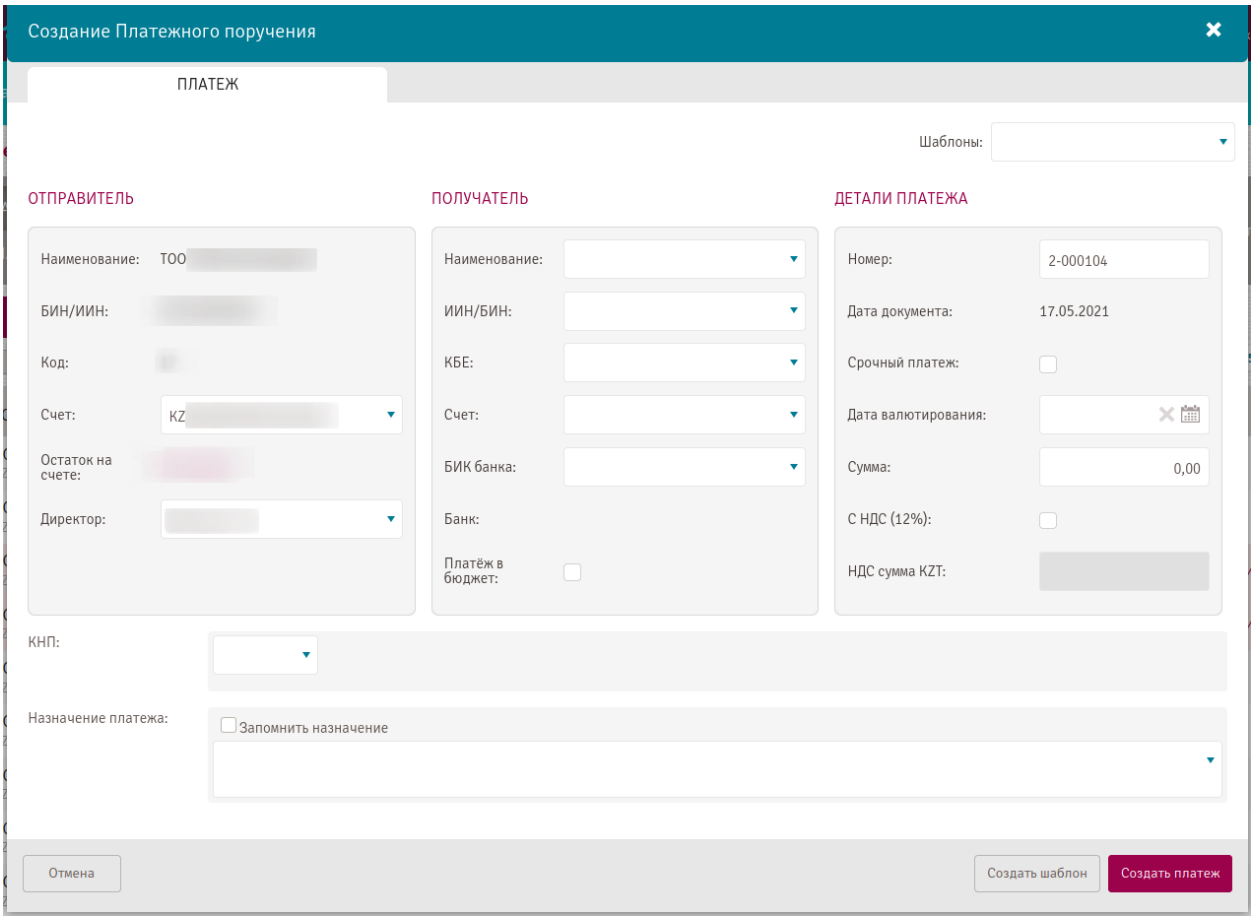

Рисунок 3.4 — Форма создания платежного поручения

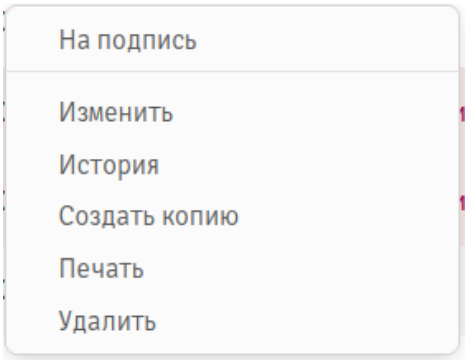

Рисунок 3.5 — Список доступных операций над документом

3.2 Интеграция с внешними системами

Для корректной работы системы необходима ее интеграция с внешними системами. В число внешних систем входят:

- процессинговая система;
- шлюз СМС-уведомлений;
- карт-процессинг.

В данном проекте была использована корпоративная сервисная шина (Enterprise Service Bus). Схема подключения шины изображена на рисунке 3.6.

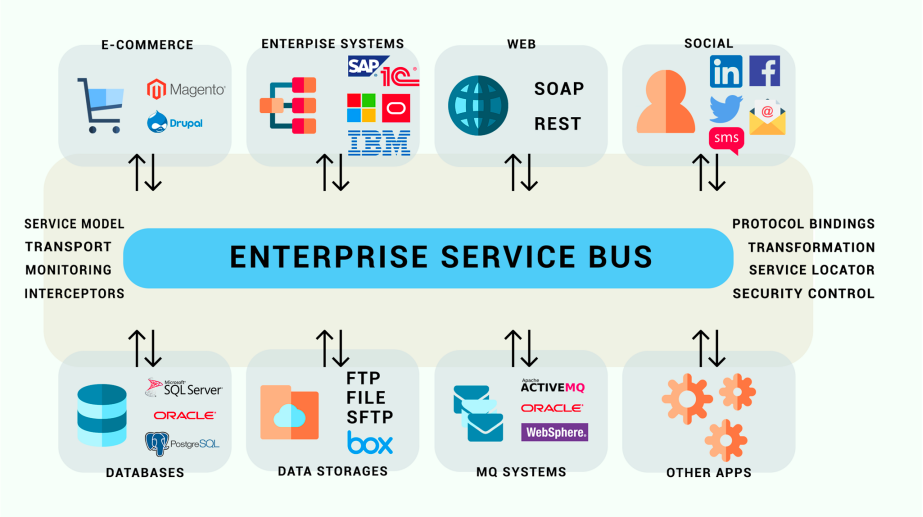

Рисунок 3.6 — Схема подключения ESB

Данный подход обладает следующими преимуществами:

1. Позволяет интегрировать несовместимые версии ПО благодаря поддержке адаптеров;
2. Гибко настраиваемая маршрутизация;

3. Централизованное управление;
4. Гарантированность доставки сообщений;
5. Возможность внедрения мониторинга для диагностики состояния системы;
6. Гарантирует безопасный канал передачи данных;
7. Предоставляет прозрачный механизм ограничения объема трафика.

3.3 Непрерывная интеграция и непрерывная поставка программного обеспечения

При разработке информационной системы, а в частности, при написании кода, были оптимизированы следующие операционные работы:

- Сборка проекта;
- Тестирование;
- Развертывание в тестовых средах;
- Развертывание в production среде.

Зачастую подобные операции требуют большого количества времени. Однако, перечисленные процессы поддаются частичной или полной автоматизации посредством CI/CD.

CI/CD (Continuous integration and Continuous delivery) — набор методик по непрерывной интеграции и непрерывной поставке (развертывания) программного обеспечения. Реализация данного процесса зависит от используемой системы контроля версий. При разработке приложения была использована DevOps-платформа (Development Operations) Gitlab[15], которая позволяет автоматизировать необходимые операционные процессы с помощью проекта Gitlab CI (Gitlab Continuous Integration).

Суть автоматизации перечисленных процессов заключается в написании специализированного файла “.gitlab-ci.yml”. Файл описывает необходимые шаги на языке разметки YAML, что позволяет унифицировать формат разметки. Пример файла непрерывной интеграции для платформы Gitlab CI приведен в Приложении В.

Результатом подобного механизма работы с кодом является ускорение разработки, тестирования и развертывания кода в различных средах, в том числе и в production-среде. В случае возникновения ошибок на одном из этапов, описанных в файле конфигурации “.gitlab-ci.yml”, разработчик имеет возможность быстрой диагностики проблемы посредством просмотра и анализа соответствующих журналов сборки.

3.4 Интеграционное и модульное тестирование

Для обеспечения стабильной работы приложения необходимо внедрение тестирования. Разделяют три разновидности тестирования кода по степени изолированности кода:

1. Unit testing (блочное тестирование). Позволяет произвести тестирование одного изолированного модуля.
2. Integration testing (интеграционное тестирование). Позволяет произвести проверку группы взаимодействующих друг с другом модулей.
3. System testing (системное тестирование). Позволяет протестировать работу системы в целом.

Внедрение тестирования в проекте зачастую сопровождается увеличением времени разработки, т.к. увеличивается объём кода (для тестирования кода необходимо написание другого кода). Однако, затраты времени на тестирование значительно сокращает затраты времени на поиск и исправление ошибок в системе в долгосрочной перспективе.

При написании дипломной работы были применены блочное и интеграционное тестирование.

Блочное тестирование подразумевает проверку работоспособности модуля в изоляции от всей системы. При этом, необходима проверка как успешных сценариев использования модуля, как и ошибочных сценариев.

Интеграционное тестирование было реализовано при помощи Selenium WebDriver (рисунок 3.7) — инструмента для автоматизации различных действий пользователя в веб-браузере.

Качество тестирования определяется степенью покрытия кода (code coverage). Данное значение исчисляется в процентах строк, которые были выполнены программой в рамках тестирования.

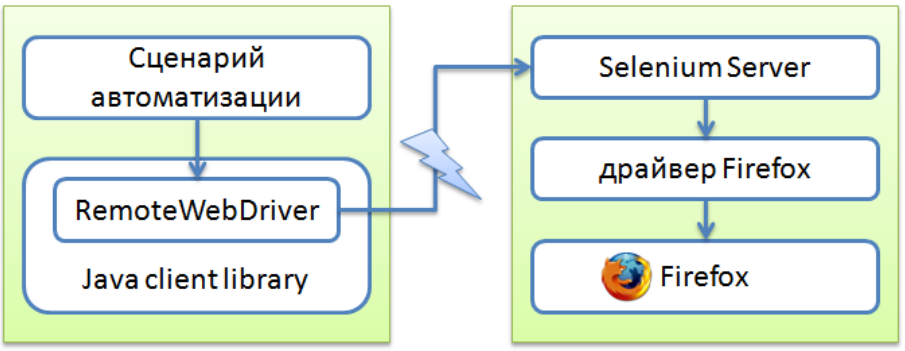

Рисунок 3.7 — Схема работы Selenium WebDriver

4 ВНЕДРЕНИЕ В ЭКСПЛУАТАЦИЮ

4.1 Эксплуатация системы

ИС переходит на фазу внедрения после обретения ею базовой функциональности. Данные этап очень важен и и позволяет определить, насколько разработанная система удовлетворяет изначально установленным требованиям.

После успешного внедрения, система переходит на этап сопровождения. Сопровождение программы — это процесс изменения системы или компонента после его поставки заказчику с целью устранения ошибок, увеличения производительности или других параметров, а также адаптация к вновь изменившимся условиям рынка.

Выделяют две основные задачи этапа сопровождения и внедрения системы:

1. Сравнение функционала системы с предыдущими итерациями разработки.
2. Настройка программного продукта.

При вводе информационной системы в эксплуатацию, зачастую требуются изменения для различных пользователей. Данные изменения направлены на оптимизацию работы в системе, а также упрощение интерфейса взаимодействия.

Модернизация является важнейшим этапом сопровождения, который определяет успех работы ИС, т.к. происходит непрерывный процесс улучшения продукта. Модернизация включает в себя расширение возможностей в функциональном плане и улучшение характеристик выполнения отдельных задач пользователей.

4.2 Пути развития

Для дальнейшего развития проекта, в первую очередь, необходима оптимизация процесса по расследования инцидентов. Внедрение Request Tracing (Open Telemetry) позволит прослеживать путь запроса от клиентского приложения и эффективнее выявлять причины неполадок.

Архитектура информационной системы была спроектирована с учетом перспективы дальнейшего разбиения сервисов на микросервисы, что

позволит, в свою очередь, горизонтально масштабировать реплики приложения для обработки большего количества запросов.

Для межсервисной коммуникации необходимо обеспечить надежный способ передачи сообщений посредством очередей. В качестве брокера сообщений в подобных системах зачастую используются:

- Apache Kafka[13]
- RabbitMQ[14]

С целью снижения излишней нагрузки на внешние сервисы (АБС Colvir) необходимо снижение запросов через корпоративную сервисную шину (Enterprise Service Bus). В качестве альтернативы ESB возможно использование проприетарного адаптера с функцией ограничения количества запросов.

Одним из путей развития бизнес-модели интернет-банкинга является оптимизация внутренних процессов, а именно — полная автоматизация документооборота.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы был исследован рынок АБС в Казахстане. Была выбрана лидирующая по числу клиентов система — Colvir, на основе которой была спроектирована и построена информационная система. В ходе выполнения поставленных задач, были изучены и проанализированы современные подходы к разработке программных продуктов. Также был разработан список требований и бизнес-правил к системе ДБО. Далее была спроектирована архитектура информационной системы и сформулированы требования к инфраструктуре банковской организации.

Были реализованы основные модули системы: модули авторизации и переводов в национальной валюте. Был настроен процесс непрерывной интеграции и доставки ПО в рамках Gitlab CI. Были реализованы адаптеры для интеграции с внешними системами. Проверка корректной работоспособности системы в целом и отдельных модулей была автоматизирована посредством интеграционных автотестов, а также модульных тестов. Спроектированная и разработанная системы бала внедрена в эксплуатацию и претерпела немало изменений в ходе активного пользования.

Были рассмотрены перспективы развития проекта с технической точки зрения для повышения производительности системы и оптимизации внутренних процессов. В результате, разработан план по улучшению системы, одним из ключевых этапов которого является избавление от корпоративной сервисной шины (ESB). Кроме того, были рассмотрены пути развития бизнес-модели проекта.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Кража денег с банковских счетов путем перехвата кодов в SMS [Электронный ресурс]. URL: https://blog.kaspersky.kz/ss7-hacked/17911/

2 Автоматизированные банковские системы [Электронный ресурс]. URL: https://www.tadviser.ru/index.php/АБС_-_Автоматизированные_банковские_системы

3 Черкасова, Е.А. Информационные технологии в банковском деле: учеб. пособие / Е.А. Черкасова, Е.В. Кийкова. – М.: Издательский центр «Академия», 2011. – 320 с.

4 Клиенты Colvir Software Solutions [Электронный ресурс]. URL: https://www.colvir.com/customers

5 Портфель корпоративных вкладов в Казахстане [Электронный ресурс]. URL: http://finprom.kz/ru/article/portfel-korporativnyh-vkladov-za-mesyac-sokratilsya-na-205-milliarda-tenge-v-plyuse-vsego-3-banka-iz-topovoj-desyatki-i-lish-7-bvu-v-celom-po-sektoru

6 Pattern: Monolithic Architecture [Электронный ресурс]. URL: https://microservices.io/patterns/monolithic.html

7 Pattern: Microservice Architecture [Электронный ресурс]. URL: https://microservices.io/patterns/microservices.html

8 A JavaScript library for building user interfaces [Электронный ресурс]. URL: https://reactjs.org/

9 CAP Theorem [Электронный ресурс]. URL: https://www.ibm.com/cloud/learn/cap-theorem

10 ACID [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/ACID

11 Ubuntu installation guide [Электронный ресурс]. URL: https://ubuntu.com/tutorials/install-ubuntu-desktop

12 SPARC T8-2 Server specification [Электронный ресурс]. URL: https://www.oracle.com/us/products/servers-storage/sparc-t8-2-ds-3864232.pdf

13 Apache Kafka [Электронный ресурс]. URL: https://kafka.apache.org/

14 RabbitMQ [Электронный ресурс]. URL: https://www.rabbitmq.com/

15 Gitlab [Электронный ресурс]. URL: https://gitlab.com

ПРИЛОЖЕНИЕ А

Oracle DDL

```
create table DOCUMENT_STATE
(
        DOCUMENT_STATE VARCHAR2(30 char) not null
        constraint PK_DOCUMENT_STATE
        primary key,
        NAME        VARCHAR2(250 char)
);
create table DOCUMENT_TYPE
(
        DOCUMENT_TYPE VARCHAR2(30 char)   not null
        constraint PK_DOCUMENT_TYPE
        primary key,
        IS_DELETED      NUMBER(1) default 0 not null
        constraint C_IS_DELETED_DOCUMENT_TYPE
        check (is_deleted in (0, 1))
);
create table CURRENCY
(
        ISO_CODE VARCHAR2(3 char) not null
        constraint PK_CURRENCY
        primary key,
        NAME        VARCHAR2(255 char),
        DIGITAL_CODE VARCHAR2(3 char)
);
create table CUSTOMER
(
        ID              NUMBER(18)                      not null
        constraint PK_CUSTOMER
        primary key,
        NAME        VARCHAR2(120 char) default NULL,
        TAX_CODE            VARCHAR2(12 char),
        DISTRICT            VARCHAR2(200 char),
        ZIP             VARCHAR2(20 char),
        CITY            VARCHAR2(200 char),
        ADDRESS             VARCHAR2(500 char) default NULL,
```

```
        REGISTERED      DATE,
        CONTRACT_STATUS VARCHAR2(10 char),
        CREATED         DATE                default sysdate not null,
        DOC_REG_PLACE   VARCHAR2(500 char),
        ORG_TYPE        VARCHAR2(100),
        OWNERSHIP_CODE  VARCHAR2(100)
);
create table ACCOUNTS
(
        ID              NUMBER(18)      not null
        constraint PK_ACCOUNTS
        primary key,
        CUSTOMER_ID   NUMBER(18)      not null
        constraint FK_ACCOUNT_CUSTOMER
        references CUSTOMER,
        ACCOUNT_NUMBER VARCHAR2(34 char) not null,
        ACCOUNT_TYPE   VARCHAR2(32 char) not null,
        CURRENCY        VARCHAR2(3 char)  not null
        constraint FK_ACCOUNT_CURRENCY
        references CURRENCY,
        OPENED          DATE,
        CLOSED     DATE,
        BALANCE         NUMBER(22, 2),
        STATUS      VARCHAR2(32 char)
);
create table DOCUMENT
(
        ID              NUMBER(18)      not null
        constraint PK_DOCUMENT
        primary key,
        DOCUMENT_TYPE   VARCHAR2(30 char) not null
        constraint FK_DOCUMENT_TYPE
        references DOCUMENT_TYPE,
        DOCUMENT_STATE  VARCHAR2(30 char) not null
        constraint FK_WORKFLOW_STATE_DOCUMENT
        references DOCUMENT_STATE,
```

```
        CUSTOMER_ID   NUMBER(18)      not null
        constraint FK_DOCUMENT_CUSTOMER
        references CUSTOMER,
        CREATED           TIMESTAMP(6),
        UPDATED           TIMESTAMP(6),
        DOCUMENT_NUMBER VARCHAR2(100 char),
        BANK_RESPONSE   VARCHAR2(2000 char) default NULL,
        INFO          VARCHAR2(420 char)
);
create table PAYMENT
(
        ID              NUMBER(18) not null
        constraint PK_PAYMENT
        primary key
        constraint FK_PAYMENT_DOCUMENT
        references DOCUMENT,
        ACCOUNT_ID              NUMBER(18)
        constraint FK_PAYMENT_ACCOUNT
        references ACCOUNTS,
        AMOUNT          NUMBER(22, 2),
        VALUE_DATE              DATE,
        PURPOSE         VARCHAR2(4000)        default NULL,
        BENEFICIARY_NAME VARCHAR2(1000)        default NULL,
        BENEFICIARY_TAX_CODE VARCHAR2(140 char),
        BENEFICIARY_ACCOUNT  VARCHAR2(34 char),
        COMMISSION              VARCHAR2(500 char) default '0'
);
create table DOMESTIC_TRANSFER
(
        ID                      NUMBER(18) not null
        constraint PK_DOMESTIC_TRANSFER
        primary key
        constraint FK_DOMESTICTRANSFER_PAYMENT
        references PAYMENT,
        BUDGET_CLASSIFICATION_CODE VARCHAR2(32 char),
        PAY_DATE                DATE);
```

ПРИЛОЖЕНИЕ Б

Модульное тестирование конкурентной обработки данных

```
package concurrency

import (
    "fmt"
    "io/ioutil"
    "math/rand"
    "net/http"
    "testing"
    "time"
)

func TestPoolDontStopOnError(t *testing.T) {
    links := []string{
        "https://golang.org/pkg/archive/",
        "https://golang.org/pkg/archive/tar/",
        "incorrect url 1",
        "https://golang.org/pkg/archive/zip/",
        "https://golang.org/pkg/bufio/",
        "https://golang.org/pkg/builtin/",
        "https://golang.org/pkg/bytes/",
        "incorrect url 2",
        "https://golang.org/pkg/compress/flate/",
        "https://golang.org/pkg/compress/zlib/",
        "incorrect url 3",
        "https://golang.org/pkg/crypto/des/",
    }

    pool := NewPool(2, false)
    for i := range links {
        i := i // introduce local variable
        task := NewTask(func() (interface{}, error) {
            resp, err := http.Get(links[i])
            if err != nil {
                return nil, err
```

```
                }
                defer func() { _ = resp.Body.Close() }()
                return ioutil.ReadAll(resp.Body)
        })
        pool.AddTask(task)
    }
    pool.Run()
    pool.Wait()
    if pool.IsCancelled() {
        t.Error("cancelled, but 'stopOnFirstError' is false")
    }
    if len(pool.TasksWithErrors()) != 3 {
        t.Error("errors are not detected")
    }
}

func TestPoolStopOnError(t *testing.T) {
    links := []string{
        "https://golang.org/pkg/archive/",
        "https://golang.org/pkg/archive/tar/",
        "incorrect url 1",
        "https://golang.org/pkg/archive/zip/",
        "https://golang.org/pkg/bufio/",
        "https://golang.org/pkg/builtin/",
        "https://golang.org/pkg/bytes/",
        "incorrect url 2",
        "https://golang.org/pkg/compress/flate/",
        "https://golang.org/pkg/compress/zlib/",
        "incorrect url 3",
        "https://golang.org/pkg/crypto/des/",
    }

    pool := NewPool(2, true)
    for i := range links {
        i := i // introduce local variable
```

```
        task := NewTask(func() (interface{}, error) {
                // User defined logic goes here
                resp, err := http.Get(links[i])
                if err != nil {
                        return nil, err
                }
                defer func() { _ = resp.Body.Close() }()
                return ioutil.ReadAll(resp.Body)
        })

        pool.AddTask(task)
    }
    pool.Run()
    pool.Wait()
    if !pool.IsCancelled() {
        t.Error("not cancelled, but 'stopOnFirstError' is true")
    }
    if len(pool.TasksWithErrors()) == 0 {
        t.Error("errors are not detected")
    }
}

func TestPoolNoErrors(t *testing.T) {
    links := []string{
        "https://golang.org/pkg/archive/",
        "https://golang.org/pkg/archive/tar/",
        "https://golang.org/pkg/archive/zip/",
        "https://golang.org/pkg/bufio/",
        "https://golang.org/pkg/builtin/",
        "https://golang.org/pkg/bytes/",
        "https://golang.org/pkg/compress/flate/",
        "https://golang.org/pkg/compress/zlib/",
        "https://golang.org/pkg/crypto/des/",
    }

    pool := NewPool(2, true)
```

```
    for i := range links {
        i := i // introduce local variable

        task := NewTask(func() (interface{}, error) {
            // User defined logic goes here
            resp, err := http.Get(links[i])
            if err != nil {
                return nil, err
            }
            defer func() { _ = resp.Body.Close() }()
            return ioutil.ReadAll(resp.Body)
        })

        pool.AddTask(task)
    }
    pool.Run()
    pool.Wait()
    if pool.IsCancelled() {
        t.Error("cancelled, but no errors expected")
    }
    if len(pool.TasksWithErrors()) > 0 {
        t.Error("errors are not detected")
    }
}

func TestPanicInTask(t *testing.T) {
    pool := NewPool(8, true)

    panicErr := fmt.Errorf("panic")

    for i := 0; i < 10; i++ {
        i := i
        pool.AddTask(NewTask(func() (_ interface{}, err error) {
            time.Sleep(time.Duration(rand.Intn(1000)) * time.Millisecond)
            if i == 5 {
                panic(panicErr)
```

```
                }
                return
        }))
    }

    pool.Run()
    pool.Wait()

    errTasks := pool.TasksWithErrors()
    if len(errTasks) == 0 {
        t.Error("0 errors, but expected 1")
    }

    if errTasks[0].Err().Error() != panicErr.Error() {
        t.Error("wrong error")
    }
}
```

ПРИЛОЖЕНИЕ В

содержимое файла .gitlab-ci.yml

```
image: golang:1.9

cache:
  paths:
        - /apt-cache
        - /go/src/github.com
        - /go/src/golang.org
        - /go/src/google.golang.org
        - /go/src/gopkg.in

stages:
  - test
  - build

before_script:
  - mkdir -p /go/src/gitlab.com/pantomath-io /go/src/_/builds
  - cp -r $CI_PROJECT_DIR /go/src/gitlab.com/pantomath-io/pantomath
  - ln -s /go/src/gitlab.com/pantomath-io /go/src/_/builds/pantomath-io
  - make dep

unit_tests:
  stage: test
  script:
        - make test

race_detector:
  stage: test
  script:
        - make race

memory_sanitizer:
  stage: test
  script:
        - make msan
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В

```
code_coverage:
  stage: test
  script:
        - make coverage

code_coverage_report:
  stage: test
  script:
        - make coverhtml
  only:
  - master

lint_code:
  stage: test
  script:
        - make lint

build:
  stage: build
  script:
        - make
```