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

Саламбаев Шынгыс Кайратулы

Разработка Интернет-банкинга для физических 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
«25» мая 2021____г.

ДИПЛОМНАЯ РАБОТА

На тему: Разработка Интернет-банкинга для

физических лиц

Специальность 5В070300 – Информационные системы

Выполнил**:** Саламбаев. Ш. К.

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

«25 » мая 2021г

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

“25” мая 2021 г.

Задание

на выполнение дипломной работы

Обучающемуся: Саламбаев Шынгыс Кайратулы
Тема: Разработка Интернет-Банкинга для Физических Лиц
Утверждена *приказом Ректора Университета №2131-б от 24.11.2020 г.*
Срок сдачи законченной работы — *18.05.2021 г.*
Исходные данные к дипломному проекту: результаты преддипломной практики, результат обзора современного состояния по данной теме, сбор теоретического материала.
Краткое содержание дипломной работы:
а) Аналитика рынка автоматизированных банковских систем;
б) Архитектура и инструменты разработки системы;
в) Разработка и тестирование;
г) Внедрение в эксплуатацию;
Рекомендуемая основная литература: *из 10 наименований*

ГРАФИК

подготовки дипломной работы

Наименования разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю**	**Примечание**
Аналитика рынка автоматизированных банковских систем	17.02.2021 г.	
Архитектура и инструменты разработки системы;	11.04.2021 г.	
Разработка и тестирование	21.04.2021 г.	
Внедрение в эксплуатацию	18.05.2021 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	20.05.2021	

Научный руководитель  Дуйсенбаева А. Н.

Задание принял к исполнению обучающийся  Саламбаев Ш. К.

Дата “ 27 ” 01 2021 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Отзыв научного руководителя

Дипломная работа

Саламбаев Шынгыс Кайратулы

Специальность 5В070300 – Информационные системы

Тема: Разработка Интернет-Банкинга для Физических 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 Саламбаев Шынгыс 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 физических лиц

Author(s)
Саламбаев Шынгыс

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	R	0
Spreads	A→	0
Micro spaces	⁞	0
White characters	▯	0
Paraphrases (SmartMarks)	a	14

Record of similarities

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

12.50% 12.50% SC1

10.86% 10.86% SC2

0.69% 0.69% QC

25
The phrase length for the SC 2

3904
Length in words

31254
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)	
1	https://coderlessons.com/tutorials/bolshie-dannye-i-analitika/vyuchit-apache-kafka/apache-kafka-klasternaia-arkhitektura	157	4.02 %
2	https://www.tadviser.ru/index.php/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B_%D0%B4%D0%B8%D1%81%D1%82%D0%B0%D0%BD%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B1%D0%B0%D0%BD%D0%BA%D0%BE%D0%B2%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D1%81%D0%BB%D1%83%D0%B6%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F	64	1.64 %
3	https://coderlessons.com/tutorials/bolshie-dannye-i-analitika/vyuchit-apache-kafka/apache-kafka-kratkoe-rukovodstvo	58	1.49 %
4	https://ekonomika-student.com/avtomatizirovannye-bankovskie-sistemy.html	57	1.46 %

АННОТАЦИЯ

Целью данной дипломной работы является разработка высоконагруженной информационной системы для предоставления доступа к дистанционному банковскому обслуживанию из любой точки, имеющей доступ в интернет, с использованием современных технологий и языков программирования.

При реализации был проведен анализ существующих систем в данной предметной области. Данная информационная система разработана на основе всех современных технологий, методологий и парадигм, таких как: микросервисная архитектура, Agile, SOLID, Domain-Driven Design (DDD), REST и другие.

Использование передовых технологий позволяет системе оставаться “гибкой” на протяжении всего жизненного цикла проекта, увеличивает скорость разработки нового функционала, оперативное реагирование на возникающие проблемы, в связи с чем пользователи получают положительный опыт от использования приложения, и, как результат, уменьшение расходов на содержание персонала в отделениях банка, увеличение клиентской базы и дохода.

THE ANNOTATION

The purpose of this thesis is to develop a highly loaded information system to provide access to remote banking services from any point with Internet access using modern technologies and programming languages.

During the implementation an analysis of existing systems in this subject area was made. This information system is developed on the basis of all modern technologies, methodologies and paradigms such as microservice architecture, Agile, SOLID, Domain-Driven Design (DDD), REST and others.

The use of advanced technologies allows the system to remain “flexible” throughout the entire project life-cycle, increases the speed of development of new functionality, rapid responses to emerging problems, and therefore users get a positive experience from using the application and as a result reduce staff costs in bank branches, increasing the client base and income.

АҢДАТПА

Осы дипломдық жұмыстың мақсаты - заманауи технологиялар мен бағдарламау тілдерін қолдана отырып, интернетке қол жетімділігі бар, кез келген нүктеден, кез келген қашықтықтағы банктік қызметтерге қол жеткізуді қамтамасыз ететін жоғары жүктелген ақпараттық жүйені дамыту.

Іске асыру барысында осы пәндік саладағы қолданыстағы жүйелерге талдау жүргізілді. Бұл ақпараттық жүйе барлық заманауи технологиялар, әдістемелер мен парадигмалар негізінде әзірленді, мысалы, микросервистік сәулет, Agile, SOLID, Domain-Driven Design (DDD), REST және басқалары.

Жетілдірілген технологияларды пайдалану жүйенің бүкіл өмірлік циклінде «икемді» болуына мүмкіндік береді, жаңа функционалдылықтың даму жылдамдығын арттырады, туындаған мәселелерге тез жауап береді, сондықтан пайдаланушылар қосымшаны қолданудың оң тәжірибесін алады, және Нәтижесінде клиенттер базасы мен кірісті көбейтіп, банк филиалдарындағы персонал шығындарын азайту.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Развитие сети Интернет и информационных технологий, а также их внедрение в повседневную жизнь, привело к необходимости получения дистанционного доступа к картам и карточным счетам и управлению ими без потребности посещения отделения банка.

Целью данной дипломной работы является предоставление пользователям доступа к банковскому обслуживанию из любой точки, имеющей доступ в интернет.

Основными задачи дипломной работы являются:

1. Интернет-банкинг. Анализ рынка автоматизированных банковских систем, а также постановка задачи дипломной работы;
2. Исследование современных технологий для разработки информационной системы;
3. Разработка архитектуры системы, с использованием изученных технологий;
4. Анализ разработанной системы и перспективы развития.

В первой главе была рассмотрена предметная область, ее функционал и автоматизированные банковские системы, предоставляемые сторонними компаниями, для автоматизации бизнес-процессов банков, а также их клиенты. На основе анализа были определены основные задачи данной дипломной работы.

Во второй главе были рассмотрены современные технологии, которые могут позволить построить высоконагруженную информационную систему, удовлетворяющую всем нынешним требованиям.

В третьей главе были разработана и описана архитектура информационной системы.

В четвертой главе был произведен анализ разработанной информационной системы, получена обратная связь от клиентов и отмечены следующие шаги развития.

1 АНАЛИТИКА РЫНКА АВТОМАТИЗИРОВАННЫХ БАНКОВСКИХ СИСТЕМ

1.1 Интернет-банкинг и его основной функционал

Интернет-банкинг - это обобщенное название технологий дистанционного банковского обслуживания. Он предоставляет доступ к счетам и операциям над ними с любой точки, имеющей доступ в сеть Интернет.

Распространение интернет-банкинга может производиться посредством “тонкого” (пользователю не требуется дополнительного программного обеспечения, информация обрабатывается интернет-браузером), а также “толстого клиента” (пользователю необходима установка дополнительного программного обеспечения: мобильного приложения и т.д.).

Как правило, услуги интернет-банкинга включают:

- выписки по счетам;
- предоставление информации по банковским продуктам;
- заявки на открытие депозитов, получение кредитов, банковских карт и т.д.;
- внутренние переводы на счета банка;
- переводы на счета в других банках;
- конвертацию средств;
- оплату услуг.

Важным свойством безопасности интернет-банкинга является подтверждение транзакций с помощью одноразовых паролей (чтобы перехват трафика не давал бы злоумышленнику возможности получить доступ к финансам). Хотя теоретическая возможность подмены сервера всё же остаётся, однако осуществление подобного мошенничества довольно проблематично.

1.2 Обзор автоматизированных банковских систем

Автоматизированная банковская система (АБС) - это комплекс технического и программного обеспечения, направленного на автоматизацию бизнес-процессов банка. Данные системы занимаются администрированием профилей клиентов, обработкой банковских операций (переводы, конвертация средств, платежи, заявки на открытие/закрытие счетов, депозитов и т.д.), документооборотом и другими процессами банка.

Архитектура систем представляет с собой модули, которые при необходимости можно редактировать или добавлять собственные, необходимые для работы индивидуальных бизнес-процессов того или иного банка и имеет следующие преимущества:

- Отсутствие дублирования данных. Информация, занесенная в один модуль, может использоваться в другом без ее ввода заново.
- Независимость модулей. Если какой-то модуль дал сбой, то его можно устранить без остановки всей системы.
- Возможность отслеживания каждой операции от начала ввода до отражения на балансе. Операции проводятся через несколько модулей, поэтому почти исключена вероятность ошибки или мошеннических действий. Все данные можно отследить в режиме реального времени.
- Корректный учет операции и формирование отчетности по требованиям Центрального Банка.
- Оперативное управление и быстрый сбор нужных данных.
- Более высокая эффективность обработки данных за счет распределения по модулям.

Основными поставщиками автоматизированных банковских систем, использующихся в Казахстане, являются: Colvir и Compass Plus. Данные системы используются на протяжении всего развития интернет-банкинга в Казахстане, так как имеют необходимый для работы функционал.

Клиентами Colvir/Compass Plus являются почти все банки Казахстана (Рисунок 1.1):

- Альфа Банк
- Алтын Банк
- Банк развития Казахстана
- Банк ЦентрКредит
- ForteBank
- Халык Банк
- АТФ Банк
- ЖССБ (ЖилСтройСберБанк)
- Жусан банк (Jýsan bank)
- KZI Bank
- Нурбанк
- Заман Банк
- Евразийский Банк Развития
- Банк Фридом Финанс Казахстан

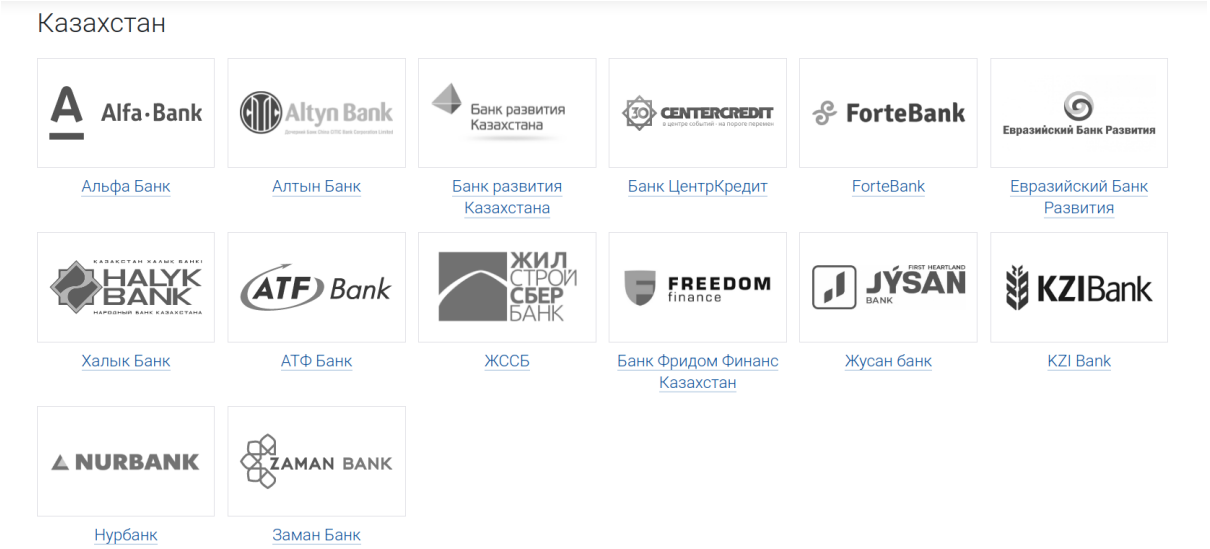

Рисунок 1.1 - Клиенты АБС Colvir

1.3. Постановка задачи

Стремительное развитие информационных технологий в Казахстане и неблагоприятная эпидемиологическая обстановка в мире привели к тому, что у пользователей появилась необходимость в дистанционном обслуживании, без физического посещения необходимых учреждений.

Основная цель дипломной работы - разработка интернет-банкинга с применением современных технологий для обслуживания клиентов и предоставления им необходимого функционала для операций над счетами, кредитами, депозитами, профилем без посещения отделений банка. На основе сформированных требований был разработан продукт, предоставляющий возможность получения всех необходимых услуг.

В результате анализа предметной области, аналогичных систем и потребностей пользователей, были сформированы следующие задачи:

- разработка архитектуры информационной системы интернет-банкинга и его основного функционала;
- интеграция с автоматизированными банковскими системами;
- реализация и запуск системы в промышленной среде.

Архитектура информационной системы будет представлена в виде схемы, с указанием всех взаимодействующих компонентов.

Следующим шагом будет реализация поставленных задач, описание процесса разработки и обоснование выбора технических инструментов.

Интеграционное тестирование с банковскими системами и проверка работоспособности.

После реализации и тестирования информационной системы, с использованием новых технологий, следует запуск в эксплуатацию.

2 АРХИТЕКТУРА И ИНСТРУМЕНТЫ РАЗРАБОТКИ

2.1 Архитектура информационной системы

Архитектура информационной системы представляет с собой концепцию, описывающую модель или структуру, связующую все компоненты системы.

При разработке информационных систем, помимо выбора используемых технологий, языка программирования и различных компонентов проекта, важна правильно разработанная архитектура, которая отвечает за расположение компонентов и их взаимодействие друг с другом.

Клиент-серверная архитектура позволяет перевести обработку и хранение данных на сервер, что в результате уменьшает нагрузку на сеть, а централизованность позволяет избежать разобщенности данных (Рисунок 2.1).

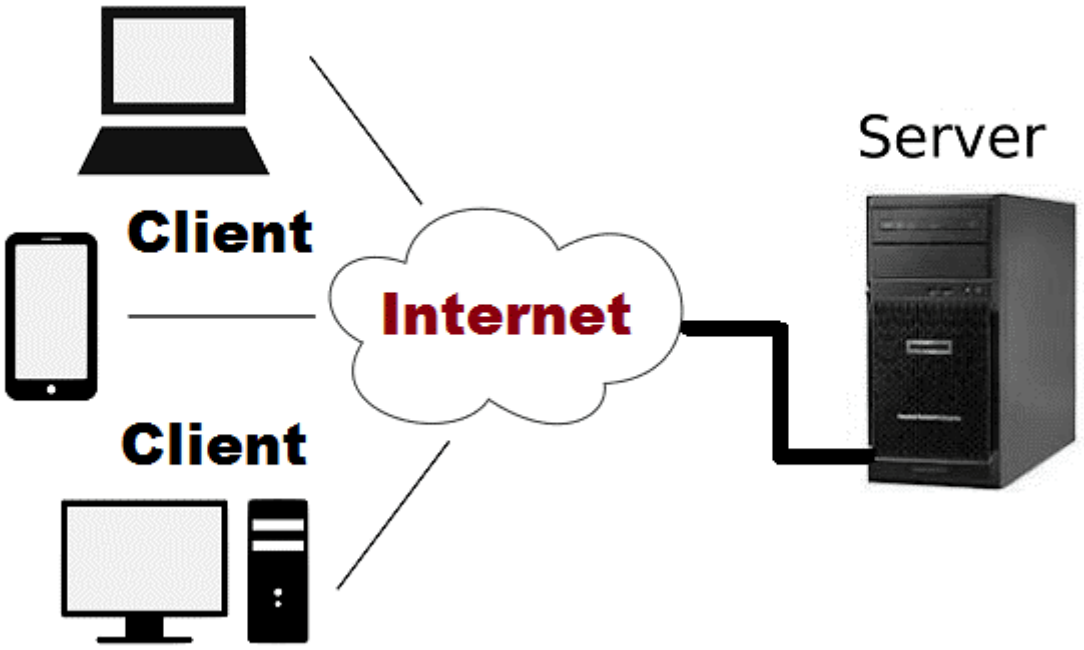

Рисунок 2.1 - Клиент-серверная архитектура

Для реализации дипломного проекта была разработана архитектура на основе микросервисов. Так информационная система содержит в себе более 100 микросервисов, которые постоянно взаимодействуют друг с другом посредствам брокера сообщений и HTTP. Диаграмма разработанной архитектура представлена на рисунке 2.2.

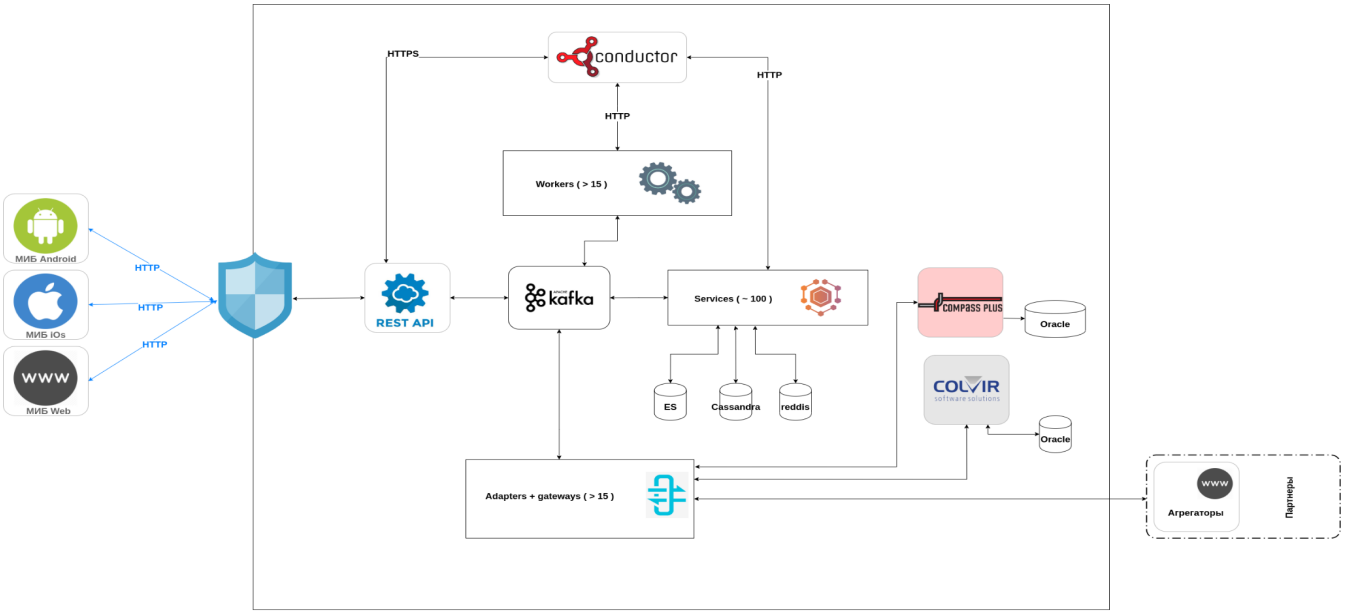

Рисунок 2.2 - Архитектура приложения дипломного проекта

Микросервисная архитектура представляет с собой архитектуру приложения, в которой представлены различные небольшие модули выполняющие свои специализированные функции и взаимодействующие между собой. В отличии от монолитной архитектуры, микросервисы имеют ряд преимуществ:

- Независимость модулей;
- Взаимозаменяемость;
- Отказоустойчивость;
- Масштабируемость;

2.2 Системы оркестрирования бизнес-процессов

При разработке информационной системы на основе микросервисов необходим инструмент, позволяющий управлять бизнес-процессами, а также выстраивать их с наименьшим изменением кода программы. Так как изменение в одном сервисе, повлечет за собой цепочку изменений в других сервисах. В небольших ИС такой подход может быть оправдан тем, что имеется небольшое количество микросервисов, взаимодействующих друг с другом, но с увеличением масштаба системы, внесение каких-либо изменений будет становится все сложнее и сложнее.

Одной из систем оркестрации бизнес-процессов (workflow) является - Netflix Conductor (Рисунок 2.3), разработанный компанией Netflix. Данная технология

позволяет расписать какой-либо процесс в формате JSON, с использованием встроенных или собственных так называемых задач (task). При изменении порядка задач или всего бизнес-процесса, изменение в коде программы, кроме самого процесса, расписанного в json файле, не требуется.

Достигается это путем формирования так называемых рабочих (worker) - небольшие микросервисы, отвечающие за определенный набор задач, который определяет пользователь (Рисунок 2.4). Таким образом возможно деление рабочих на контексты использования. Также рабочие могут обрабатывать задачи, без участия основных сервисов обработки данных, но такой подход является анти паттерном архитектуры приложения.

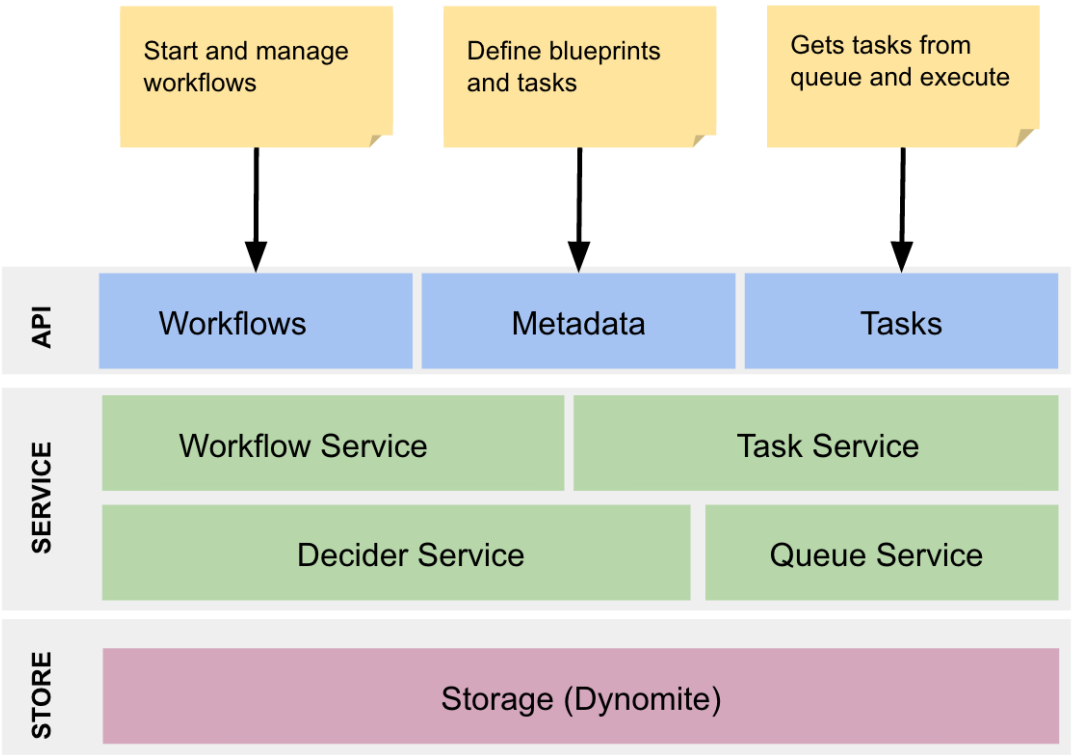

Рисунок 2.3 - Архитектура Netflix Conductor

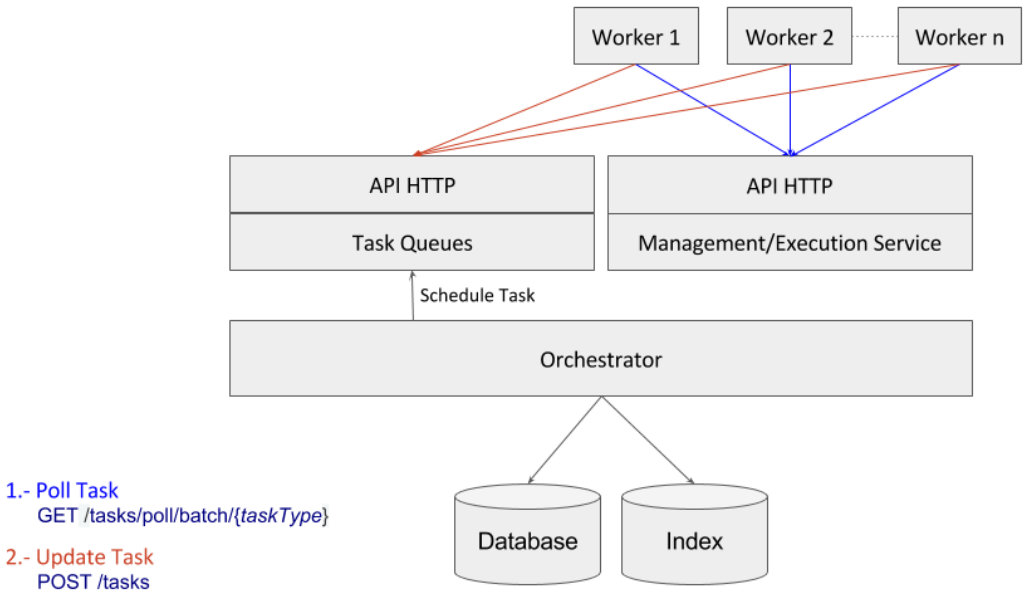

Рисунок 2.4 - Взаимодействие “рабочих” и Netflix Conductor

Распишем для примера задачу получения карт (Рисунок 2.5), на вход принимается идентификатор пользователя. Результатом будет список карт пользователя, идентификатор которого передали в задачу.

```
{
  "name": "Cards",
  "taskReferenceName": "Cards",
  "inputParameters": {
    "profileId": "${workflow.input.entity.profileId}"
  },
  "type": "SIMPLE",
  "startDelay": 0,
  "optional": false,
  "asyncComplete": false
}
```

Рисунок 2.5 - Пример задачи (task)

Таких базовых задач может быть огромное количество, которые можно переиспользовать в различных бизнес-процессах. В случае необходимости изменения бизнес-логики отпадает необходимость в изменении кода программы, так как достаточно внести изменения в сам бизнес-процесс.

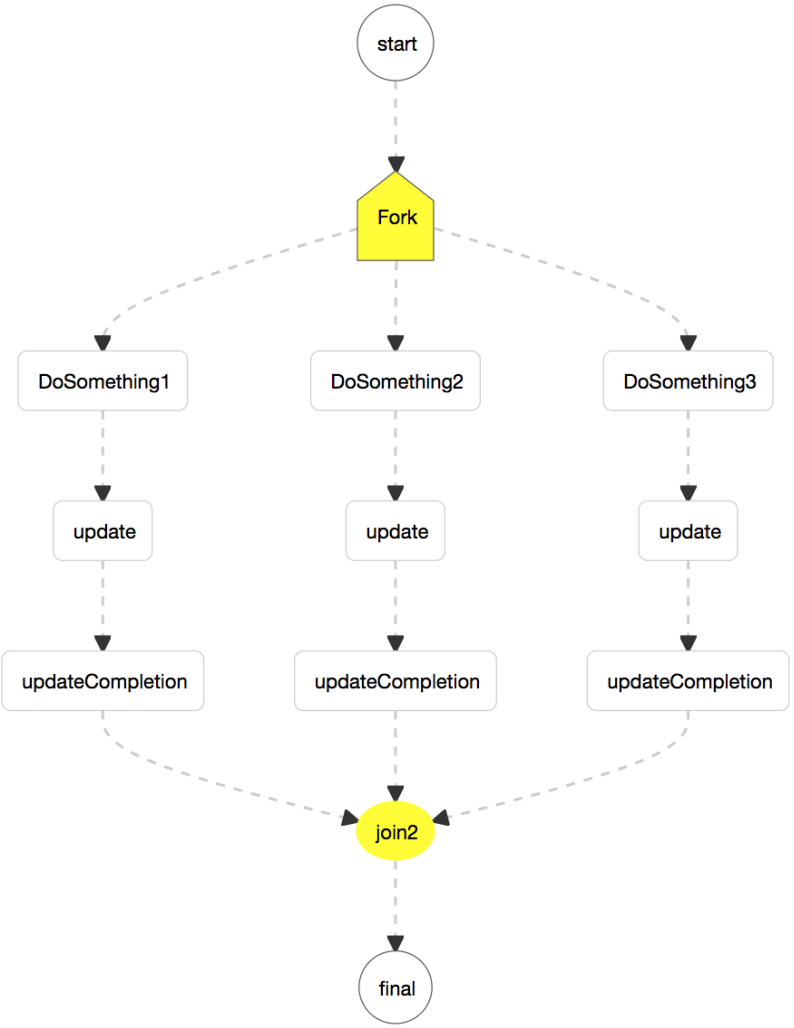

Рисунок 2.6 - Пример бизнес-процесса (workflow)

2.3 Инструменты разработки

Выбор языка программирования является важной составляющей разработки, так как на основе выбранного языка, будут выбираться все компоненты информационной системы. При разработке данной работы был выбран - Scala. Scala - достаточно молодой мультипарадигменный язык программирования, который сочетает в себе функциональное и объектно-ориентированное программирование. Язык спроектирован для быстрого создания компонентного программного обеспечения, поэтому его система типов является безопасной и исключена возможность получения ошибки типизации. Немаловажным будет отметить, что способствует этому статическая типизация.

Scala был разработан в 2003-2004 годах, однако стремительное развитие началось в начале 2010-х годов, после того как передовые компании в ИТ-сфере начали применять его в промышленной среде, а именно:

- Twitter
- Coursera
- Apache (Spark, Ignite, Kafka)

Еще одной причиной при выборе Scala стало то, что язык программирования является Java-подобным и работает на Java Virtual Machine (JVM). Это означает, что все все компоненты, технологии и код, разработанные на Java, могут исполняться и использоваться при разработке на Scala.

Java - это огромная экосистема, в которой имеется разнообразие проектов с открытым исходным кодом, программных платформ и API, разработанных не только коммерческими компаниями, но и самим сообществом. Невзирая на повсеместность Java, критика в отношении данного языка не утихает, а обсуждения сообществом аспектов языка приводит в неразрешимым спорам. Один из частых пунктов критики является синтаксис, который называют “многословным”. Scala устраняет недостатки Java, связанные не только с синтаксисом, но и с технологической составляющей, такой как масштабирование, при этом оставляя возможность использования всех функциональных возможностей Java.

Для автоматической сборки проектов, написанных на Scala используется Scala Build Tool (SBT). Результатом сборки, так же, как и в Java, является исполняемый файл в формате jar, и может быть запущен на JVM. При желании имеется возможность использовать сборщик Java - Maven, так как имеется обратная совместимость со всем, что касается Java SDK.

В качестве фреймворка был выбран Akka. Данный фреймворк когда был частью самого языка Scala, однако, в середине 2000-х годов было решено перенести его в отдельный коммерческий проект с открытым исходным кодом. Akka представляет с собой набор инструментов для реализации и работы с, так называемой, моделью акторов.

Модель акторов - математическая модель параллельных вычислений, строящаяся вокруг актора (действующее лицо). Взаимодействие между акторами происходит путем обмена сообщениями. Каждый актор инкапсулирован, имеет независимое состояние и три реакции на полученное сообщение:

- отправить N сообщений другим акторам;
- породить M новых акторов;
- изменить внутреннее состояние;

При этом нет никакой определенной последовательности вышеописанных действий, все они могут производиться параллельно.

Архитектура акторной системы, представленной в фреймворке Akka представляет с собой 3 основных уровня (Рисунок 2.7):

- Корневой
- Системная
- Пользовательский

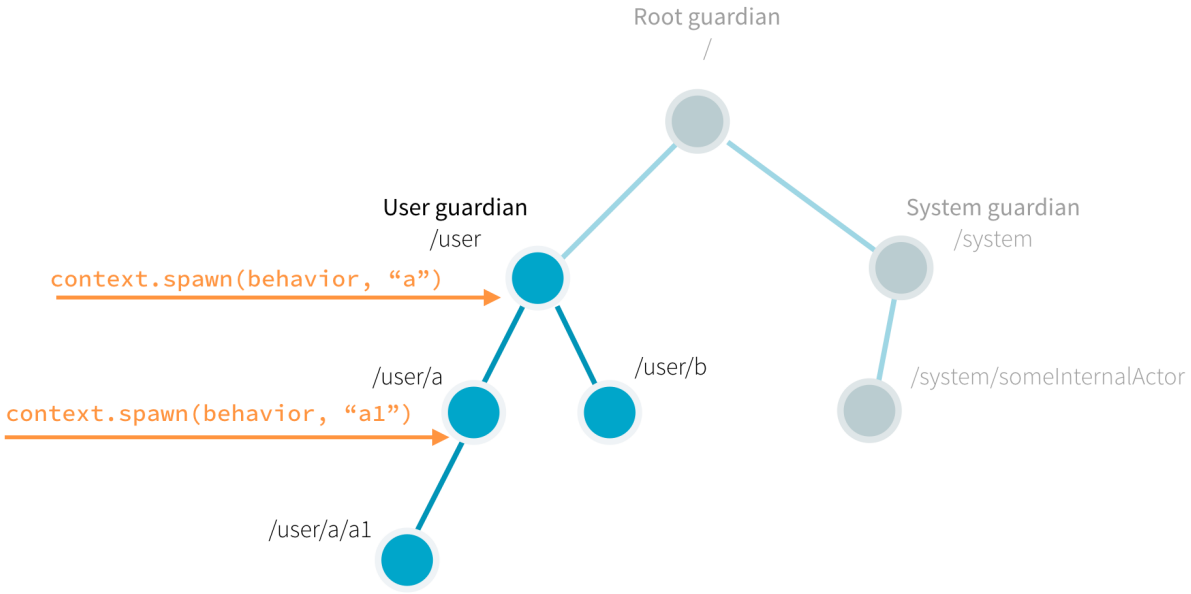

Рисунок 2.7 - Архитектура системы акторов

Корневой уровень - это родительский элемент для всех субъектов в системе, который останавливается последним при завершении работы самой системы.

Системный уровень - Akka или другие библиотеки, построенные на основе Akka, могут создавать субъекты в системном пространстве имен.

Пользовательский уровень - это актор верхнего уровня, который предоставляется для запуска всех других акторов в приложении.

Важное различие между передачей сообщений и методами вызова заключается в том, что сообщения не имеют возвращаемого значения. Отправляя сообщение, актер делегирует работу другому актеру. Если он ожидал возвращаемого значения, отправляющему актору нужно было бы либо заблокировать, либо выполнить работу другого актора в том же потоке. Вместо этого принимающий актор доставляет результаты в ответном сообщении (Рисунок 2.8).

Вторым ключевым разницей является - инкапсуляция. Акторы реагируют на сообщения так же, как объекты реагируют на вызванные к ним методы. Разница в том, что акторы выполняются независимо от отправителей сообщения и реагируют на входящие сообщения последовательно, по одному за раз. В то время как актор обрабатывает сообщения, отправленные ему последовательно, разные акторы работают одновременно друг с другом, так что система акторов может обрабатывать столько сообщений одновременно, сколько поддерживает оборудование.

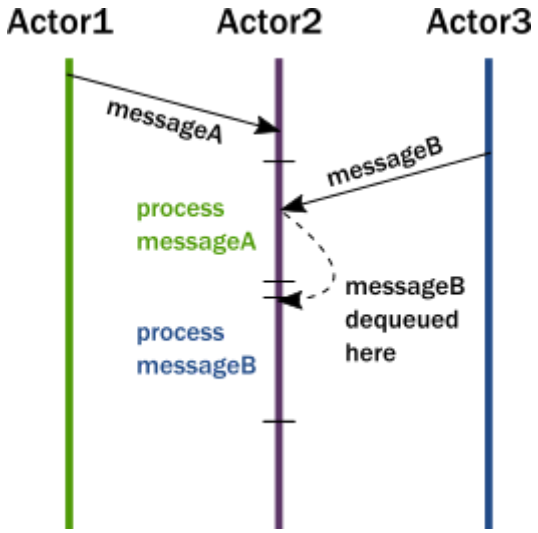

Рисунок 2.8 - Обмен сообщениями

Алгоритм работы актора при получении сообщения:

- Актор добавляет сообщение в конец очереди;
- Если актор не был запланирован для выполнения, он помечается как готовый к выполнению;
- Планировщик запускает обработку сообщения актором;
- Актор выбирает сообщение в начале очереди;
- Актор изменяет внутреннее состояние/отправляет сообщения другим актерам;
- Актор помечается как внеплановый (готовым к обработке следующего сообщения).

Для работы данного алгоритма обработки сообщений, акторы имеют:

- Почтовый ящик (очередь, в которую попадают сообщения);
- Поведение (состояние актера, внутренние переменные и т.д.);

- Сообщения (фрагменты данных, представляющие сигнал, аналогичные вызовам методов и их параметрам);
- Среда выполнения (механизм, который принимает акторов, у которых есть сообщения, для реагирования, и вызывает их код обработки сообщений);
- Адрес актора;

Сообщения попадают в почтовые ящики акторов. Поведение актора описывает, как он реагирует на сообщения (например, отправляет больше сообщений или меняет состояние). Среда выполнения создает контекст выполнения для управления всеми этими действиями.

2.4 Базы данных

Сейчас вся информация, которую генерирует каждый человек, имеющий доступ в интернет, увеличивается с каждым днем и ее необходимо хранить. Особенно потеря информации недопустима при работе с финансами, как в нашем случае. Потеря даже одной записи может повлечь за собой цепочку необратимых финансовых потерь. Исходя из всех вышеуказанных факторов было решено использовать несколько различных баз данных для увеличения отказоустойчивости и снижения рисков потери информации.

Были выбраны следующие NoSQL СУБД:

- ElasticSearch
- Apache Cassandra
- Redis

Elasticsearch (ES) – масштабируемое нереляционное хранилище данных, обладающее широким набором функций полнотекстового поиска. Преимуществом данной базы данных является то, что она имеет поисковой движок, позволяющий производить горизонтальный масштабируемый поиск с поддержкой многопоточности. Система основана на библиотеке Apache Lucene, которая предназначена для индексирования и поиска информации в любом типе документов.

В больших масштабах несколько копий ElasticSearch могут объединяться в кластер. Имеется балансировщик и координатор, отвечающий за распределение нагрузки между репликами базы данных, маршрутизацию и синхронизацию данных. Kibana - программная панель визуализации данных, хранимых в ElasticSearch. Kibana поставляется вместе с ElasticSearch и имеет интерфейсы для запросов в формате JSON в базу данных в реальном времени. Также имеется

функционал для визуализации данных в различных форматах: графики, диаграммы и т.д. На рисунке 2.9 представлена архитектура кластера ElasticSearch.

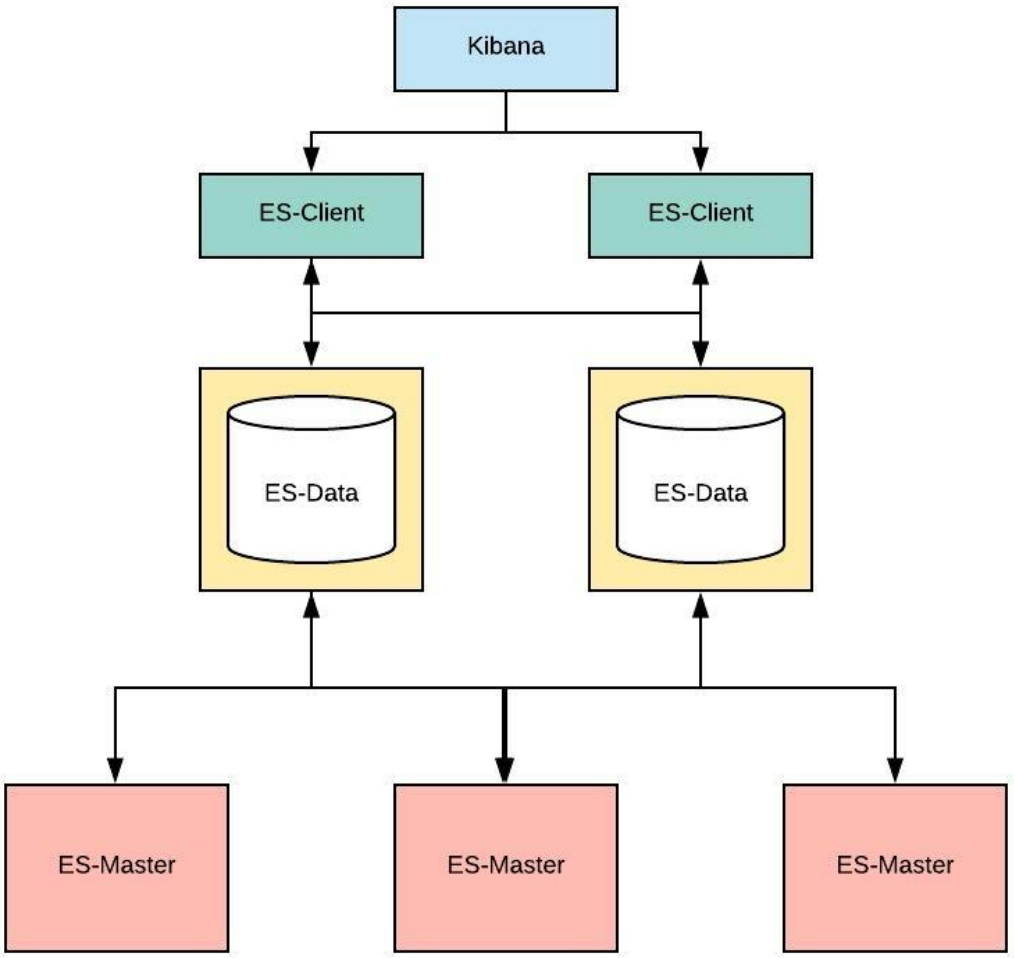

Рисунок 2.9 - Архитектура кластера ElasticSearch

Redis - система управления базами данных класса NoSQL, хранящая данные в оперативной памяти. Хранит и работает данная СУБД с данными типа ключ-значение. Данная СУБД была выбрана исключительно для хранения часто читаемых, нечувствительных данных, то есть кэша. К таким данным относится, например, состояние счета, при отсутствии операций над ним, так как клиент может часто просматривать состояние счета, но не производить никаких операций. В таком случае данные будут браться из redis, данные которого хранятся в оперативной памяти, что в результате снизит нагрузку на всю систему и увеличит скорость обработки запроса.

Apache Cassandra - это распределенная СУБД класса NoSQL с открытым исходным кодом. Данная СУБД рассчитана на создание высоко масштабируемых и

надежных хранилищ огромных массивов данных. Особенностью Cassandra можно назвать собственный язык запросов - CQL (Cassandra Query Language). Несмотря на класс NoSQL, к которому относится база данных, данные помещаются в таблицы, содержащие строки и столбцы. По этой причине CQL очень похож на SQL и применение терминов из SQL к Cassandra не будет ошибочным, но с некоторыми оговорками. Однако, отличием от SQL баз данных является то, что Cassandra является колоночной, а SQL - построчной. Cassandra так же, как и ElasticSearch, поддерживает кластеризацию.

Цель использования Cassandra и ElasticSearch - хранение данных, обе СУБД хранят идентичные данные. Такое решение было принято для увеличения отказоустойчивости всей системы в общем. При выходе из строя одной из СУБД другая может ее заменить.

Необходимо выделить то, что чтение данных происходит из двух СУБД одновременно, но результатом выполнения будет первый вернувшийся результат (Рисунок 2.10).

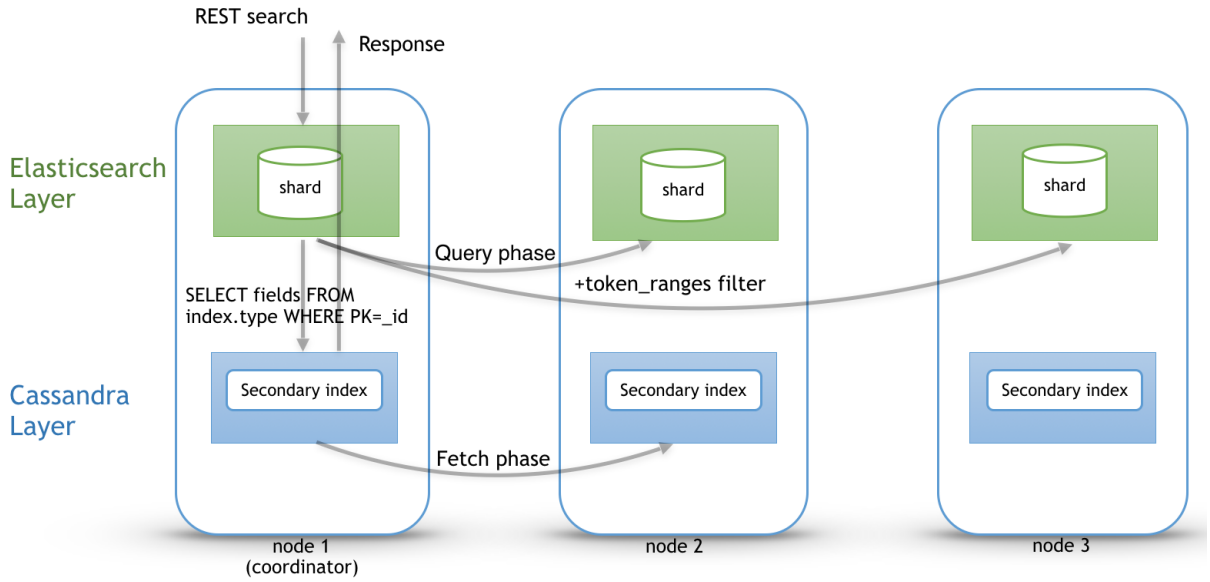

Рисунок 2.10 - СУБД. Чтение данных

3 РАЗРАБОТКА И ТЕСТИРОВАНИЕ

3.1 Среда разработки

Для разработки любого программного обеспечения требуется единая среда разработки (IDE). IDE представляет собой комплекс программных средств, упрощающих процесс разработки различного ПО. Стандартная IDE включает в себя:

- Транслятор (компилятор и/или интерпретатор);
- текстовый редактор;
- средства автоматизации сборки;
- отладчик.

Во всех новых IDE имеется поддержка система контроля версий, например, git. Имеется огромное количество IDE, универсальных и предназначенных для определенных языков. Одним из крупных поставщиков IDE является компания - JetBrains. В ассортименте предоставляемых IDE имеются: PyCharm, GoLand, PhpStorm и другие.

IntelliJ IDEA - среда разработки для JVM. Изначально, создавалась для разработки на языке Java, однако, с появлением большого количества языков программирования, работающих с Java SDK, фокус был смещен с языка на платформу виртуальную машину Java.

Говоря о Scala, IntelliJ IDEA имеет плагин, позволяющий использовать данный язык. Также имеется поддержка сборщика проектов Scala - SBT. А в недавнем обновлении было добавлено автодополнение на основе алгоритмов машинного обучения.

Также немаловажным фактом является то, что данная среда разработки имеет функционал для работы с базами данных. А умное дополнение кода, которое предлагает необходимые элементы кода, которые имеются только в конкретном контексте использования, позволяет ускорить процесс разработки и улучшить качество кода.

3.2 Брокер сообщений

Так как была выбрана микросервисная архитектура, сервисам необходимо обмениваться информацией между собой. Именно для таких задач и существуют брокеры сообщений.

Выбран был брокер сообщений - Apache Kafka, причинами данного выбора послужил:

- Надежность;
- Масштабируемость;
- Долговечность;
- Производительность;

Apache Kafka — это распределенная система обмена сообщениями «публикация-подписка» и надежная очередь, которая может обрабатывать большой объем данных и позволяет передавать сообщения из одной конечной точки в другую. Кафка подходит как для автономного, так и для онлайн-рассылки сообщений. Сообщения Kafka сохраняются на диске и реплицируются в кластере для предотвращения потери данных. Kafka построен поверх службы синхронизации ZooKeeper (Рисунок 3.1).

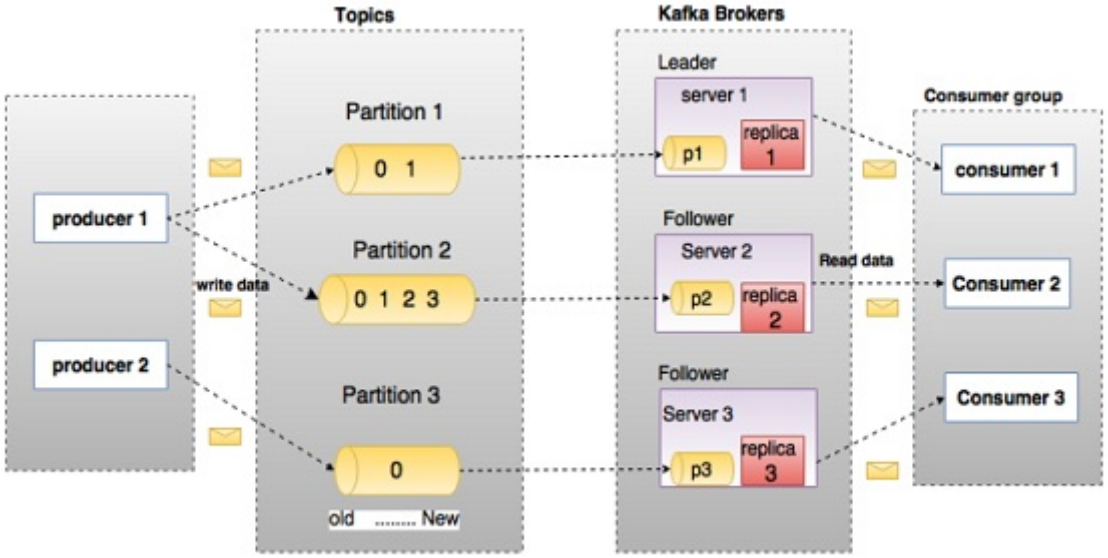

Рисунок 3.1 - Схема передачи сообщений

На следующей картинке показана кластерная диаграмма:

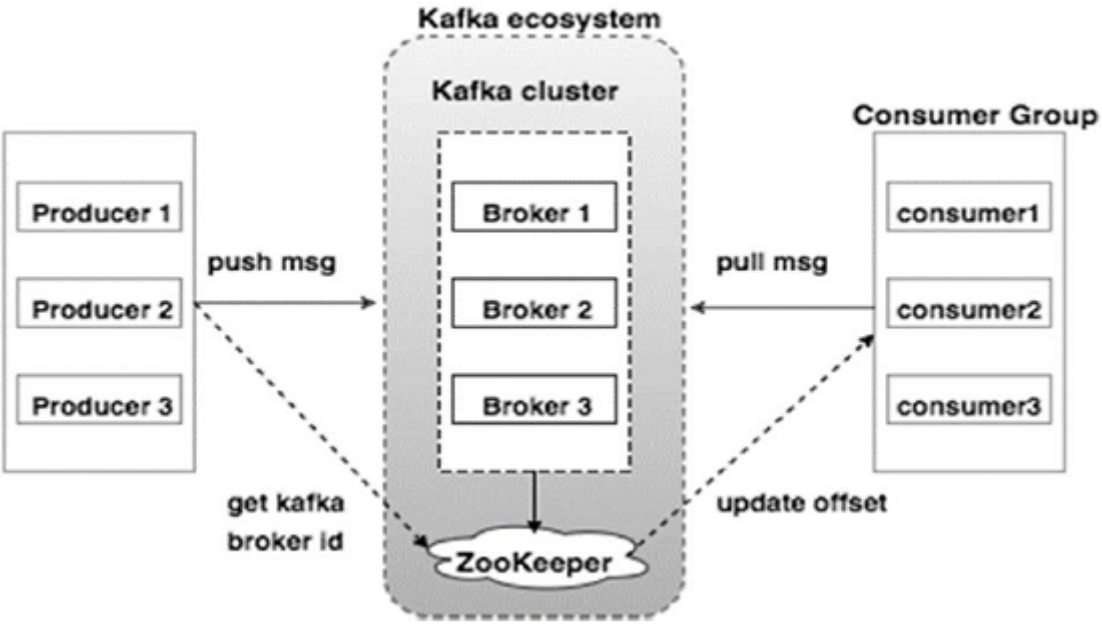

Рисунок 3.2 - Кластерная диаграмма Kafka

Кластер Kafka обычно состоит из нескольких брокеров для поддержания баланса нагрузки. Брокеры Kafka не имеют статуса, поэтому они используют ZooKeeper для поддержания своего кластерного состояния. Один экземпляр брокера Kafka может обрабатывать сотни тысяч операций чтения и записи в секунду, а каждый брокер может обрабатывать ТБ сообщений без влияния на производительность. Выбор лидера брокера Kafka может быть сделан ZooKeeper.

ZooKeeper используется для управления и координации брокера Kafka. Сервис ZooKeeper в основном используется для уведомления производителя и потребителя о наличии любого нового брокера в системе Kafka или сбое брокера в системе Kafka. В соответствии с уведомлением, полученным Zookeeper относительно присутствия или отказа брокера, производитель и потребитель принимают решение и начинают согласовывать свою задачу с каким-либо другим брокером.

Производители передают данные брокерам. Когда запускается новый брокер, все производители ищут его и автоматически отправляют сообщение этому новому брокеру. Производитель Kafka не ждет подтверждений от брокера и отправляет сообщения так быстро, как может обработать брокер.

Поскольку брокеры Kafka не имеют состояния, это означает, что потребитель должен поддерживать количество сообщений, использованных с помощью смещения раздела. Если потребитель подтверждает конкретное смещение сообщения, это означает, что потребитель использовал все предыдущие

сообщения. Потребитель отправляет брокеру асинхронный запрос на получение, чтобы подготовить буфер байтов к использованию. Потребители могут перемотать или перейти к любой точке раздела, просто указав значение смещения. Значение смещения потребителя сообщается ZooKeeper.

3.3 Описание страниц перехода

Первым, что увидит пользователь при входе в приложение, страницу для неавторизованных пользователей (Рисунок 3.1):

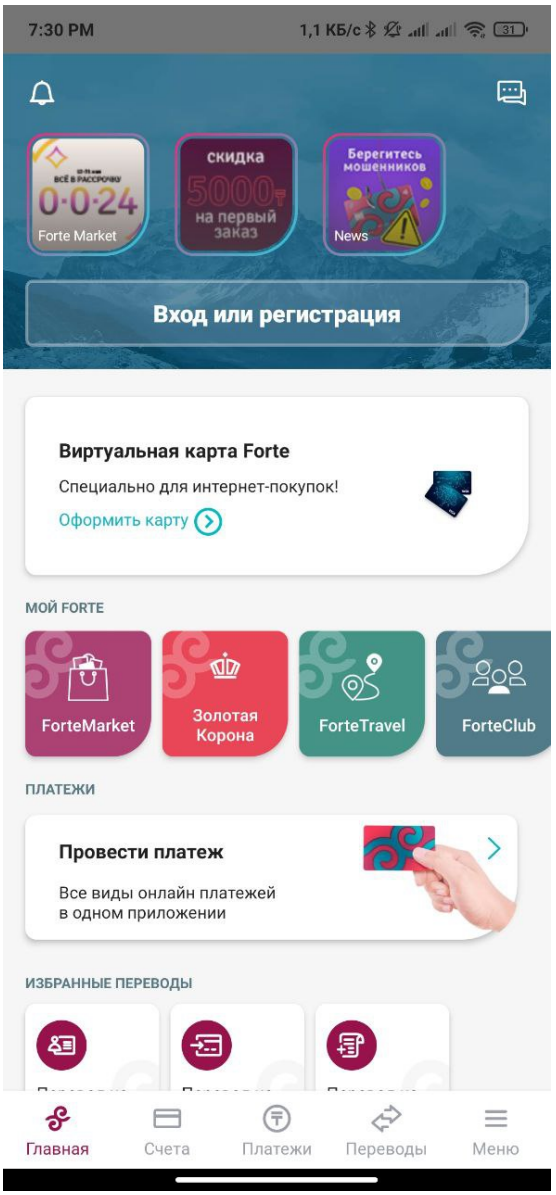

Рисунок 3.3 - Главная страница неавторизованных пользователей

Чтобы авторизоваться пользователь может нажать на кнопку “Вход или регистрация” и пройти авторизацию или регистрацию, в случае, если пользователь не был ранее зарегистрирован.

Для авторизации используется двухфакторная аутентификация, где первым этапом является получение временного пароля (One Time Password - OTP), вторым - введение пароля от аккаунта, заведенного при регистрации. На рисунках 3.2 показаны этапы авторизации:

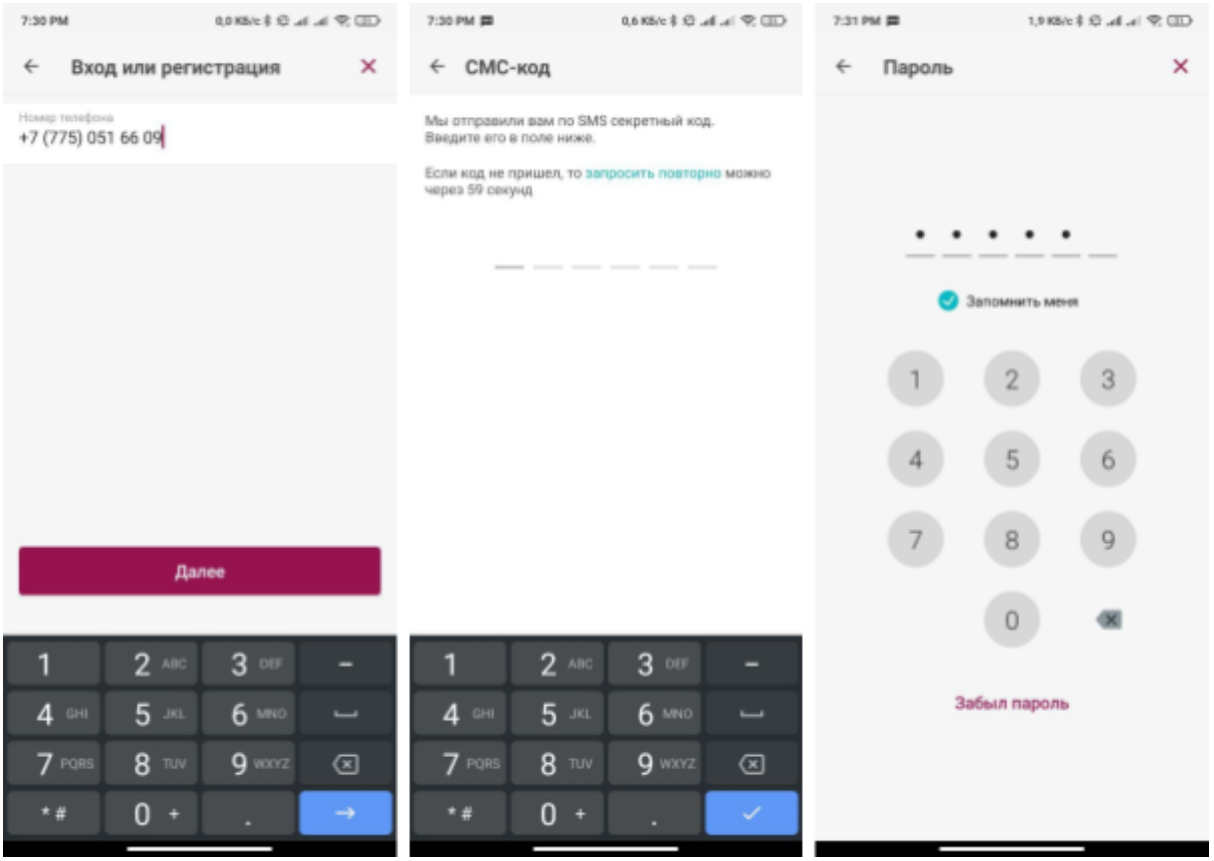

Рисунок 3.4 - Этапы авторизации

Далее авторизованный пользователь попадает на главные экран, на котором он видит свои счета, карты, избранные платежи, а также различную информацию, связанную с новостями и акциями банка (Рисунок 3.3):

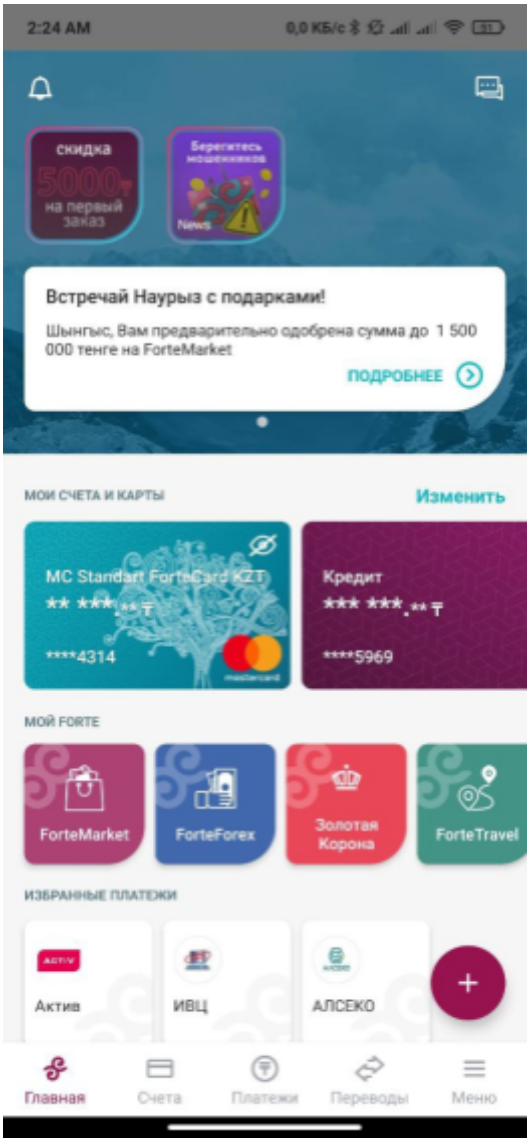

Рисунок 3.5 - Главная страница авторизованного пользователя

Также имеются различные вкладки, характерные для интернет-банкинга, а именно: Счета и карты, Платежи, Переводы и Меню, предназначенное для настроек и управления аккаунтом и приложением.

Во вкладке “Счета и карты” указана информация о различных счетах пользователя, а также ссылки на открытие новых продуктов банка.

Во вкладке “Платежи” показываются различные поставщики, услуги которых пользователь может оплатить.

Во вкладке “Переводы” имеется возможность выбора нужного типа перевода. Например, на карту или по номеру телефона пользователя банка (Рисунок 3.4).

Во вкладке “Меню” можно изменить настройки не только профиля, но и самого приложения (Рисунок 3.5).

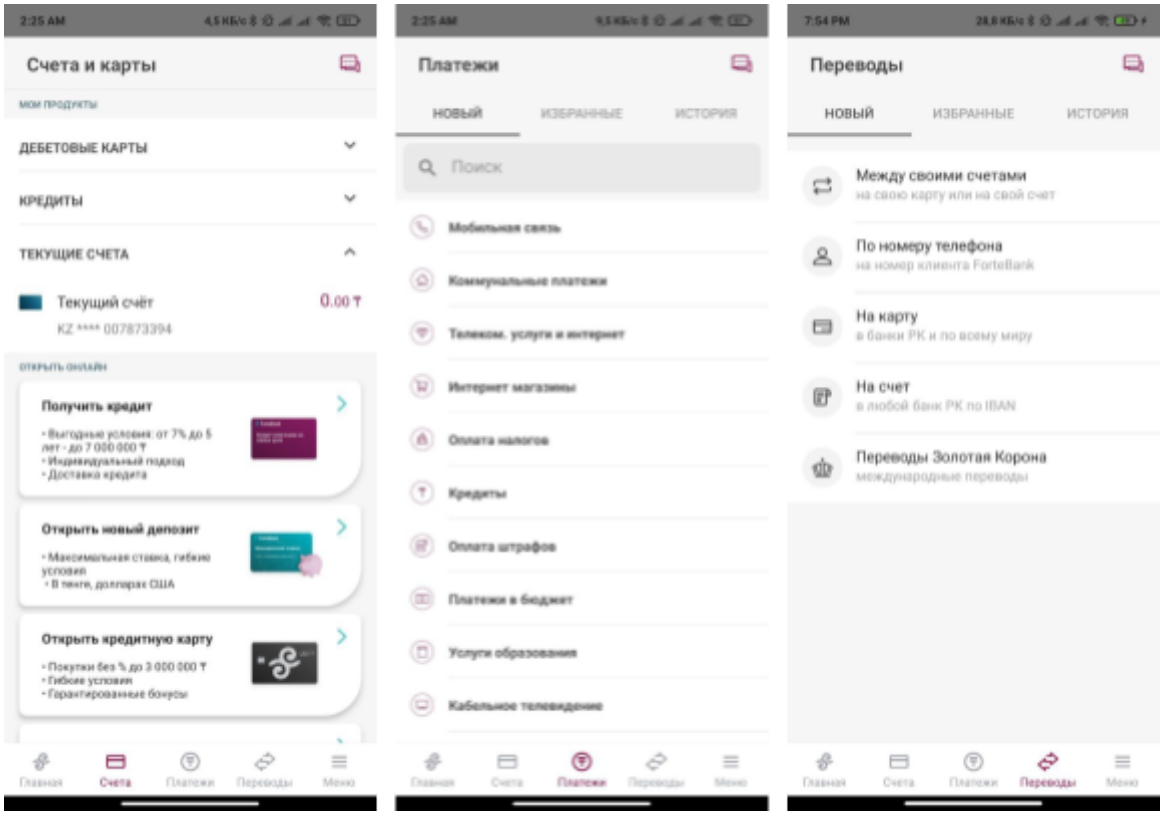

Рисунок 3.6 - Вкладки “Счета и карты”, “Платежи”, “Переводы”

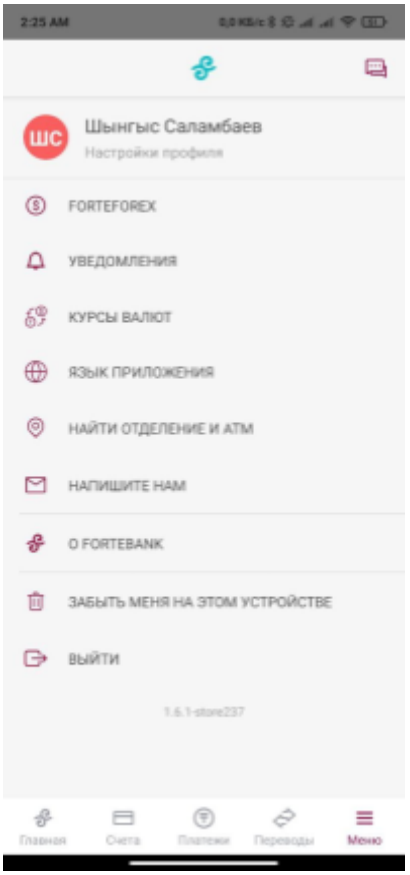

Рисунок 3.7 - Вкладка “Меню”

4 ВВЕДЕНИЕ В ЭКСПЛУАТАЦИЮ

4.1 Эксплуатация системы

Особенной возможностью разработанного интернет-банкинга является получение онлайн-справок. Данные справки имеют печать банка и являются официальным документом.

Данные справка удостоверяет наличие определенного счета или карты и карточного счета у пользователя. Еще одной особенностью является то, что данную справку возможно получить на трех основных языках: русском, казахском, английском. Языки выбирают с помощью checkbox, формат файла сформированной справки - pdf. При выборе всех или нескольких языков в одном файле формируются соответственное количество страниц, на каждой из которых находится информация на определенном языке (Рисунок 4.1).

Также имеется возможность выбора указания остатка на счету, при выборе пункта “С указанием остатка” будет сформирована справка с информацией об остатке средств на счете в виде таблицы. К тому же информация об оставшихся средствах будет переведена на другие основные валюты, такие как: Евродоллар и Доллар США, с указанием текущего курса валют, а в отдельном столбце все суммы будут указаны прописью.

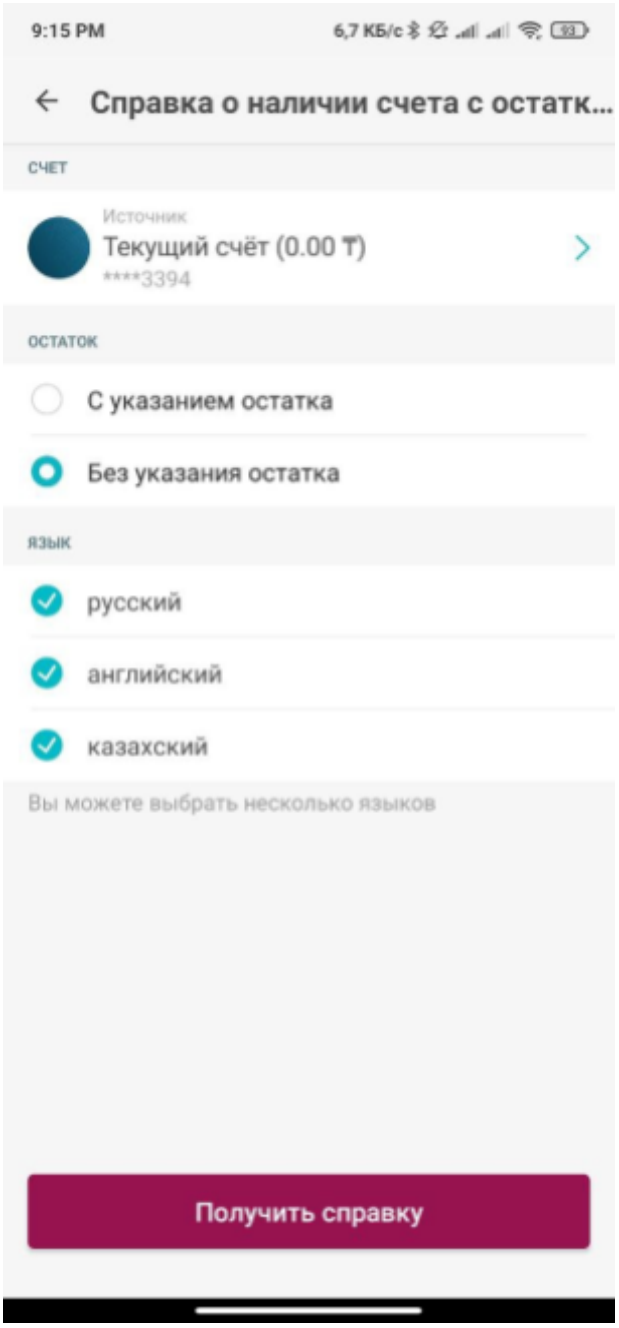

Рисунок 4.1 - Меню выбора настроек онлайн-справки

ForteBank

№ 103147863185223064 6

Дата 15.05.2021 21:31 (по времени города Нур-Султан)

Акционерное общество «ForteBank»

БИК: IRTYKZKA

БИН: 990740000683

СПРАВКА

Настоящим АО "ForteBank" подтверждает, что Саламбаев Шынгыс Кайратулы, документ удостоверяющий личность №038486263, выдан 27/08/2015 года, МВД РК, ИИН 981221300126, имеет счет:

Номер счета	Сумма в валюте счета	Сумма прописью	Валюта счета	Тип счета
	0.00	Ноль тенге 00 тиын	KZT	Текущий счет
	Сумма в эквиваленте	Сумма прописью	Валюта эквивалента	Курс НБ РК
KZ9496500F0007873394	0.00	Ноль долларов США 00 центов	USD	428.71
	Сумма в эквиваленте	Сумма прописью	Валюта эквивалента	Курс НБ РК
	0.00	Ноль евро 00 евроцентов	EUR	519.12

Проверить подлинность электронного документа Вы можете по ссылке QR код

Рисунок 4.2 - Справка на русском языке

ForteBank

№ 7663448578788576317

Date 15.05.2021 21:31 (Nur-Sultan local time)

ForteBank Joint-stock company

BIC: IRTYKZKA

BIN: 990740000683

Certificate

ForteBank JSC herewith confirms that Salambaev Shyngys Kajratuly, identity document No.038486263 dated 27/08/2015 issued by the MIA RK, IIN 981221300126, has the following account:

Account number	Amount in the account currency	Amount in letters	Account currency	Account type
	0.00	Zero tenge 00 Tyn	KZT	Current account
	Equivalent amount	Amount in letters	Equivalent currency	Exchange rate of the NBR
KZ9496500F0007873394	0.00	Zero US dollars 00 Cents	USD	428.71
	Equivalent amount	Amount in letters	Equivalent currency	Exchange rate of the NBR
	0.00	Zero EURO 00 Eurocents	EUR	519.12

You may verify the authenticity of the electronic document by following the link QR code

Рисунок 4.3 - Справка на английском языке

ForteBank

№ 7903210492183072341

Күні 15.05.2021 21:31 (Нұр-Сұлтан қаласының уақыты бойынша)

«ForteBank» акционерлік қоғамы

БСК: IRTYKZKA

БИН: 990740000683

Анықтама

"ForteBank" АҚ осы арқылы Саламбаев Шынғыс Кайратулы, жеке басын куәландыратын құжаты №038486263, 27/08 /2015ж. ҚР ІІМ берген, ЖСН 981221300126, мынадай шотының бар екенін растайды:

Шот нөмірі	Шот валютасындағы сома	Жазбаша түрде сома	Шот валютасы	Шот түрі
	0.00	Ноль теңге 00 тиын	KZT	Ағымдағы шот
	Балама сома	Жазбаша түрде сома	Балама валюта	ҚР ҰБ бағамы
KZ9496500F0007873394	0.00	Ноль АҚШ доллары 00 цент	USD	428.71
	Балама сома	Жазбаша түрде сома	Балама валюта	ҚР ҰБ бағамы
	0.00	Ноль еуро 00 евроцент	EUR	519.12

Электронды құжаттың түпнұсқалығын QR код сілтемесі бойынша тексере аласыз

Рисунок 4.4 - Справка на казахском языке

На рисунках 4.2 - 4.4 показаны сформированные справки на различных языках с указанием остатка на счете.

Также в каждой сформированной справке имеется QR-код, который содержит в себе информацию об этой справке. QR-код может считать любой человек, которому необходимо удостовериться в подлинности справки, при считывании, пользователь будет перенаправлен на сайт, на котором будет показана сформированная и выданная справка. Это сделано на случай, если информация в справке будет кем-либо изменена или подделана, в таком случае при считывании QR-кода и получении подлинной справки по данной ссылке можно удостовериться в достоверности информации.

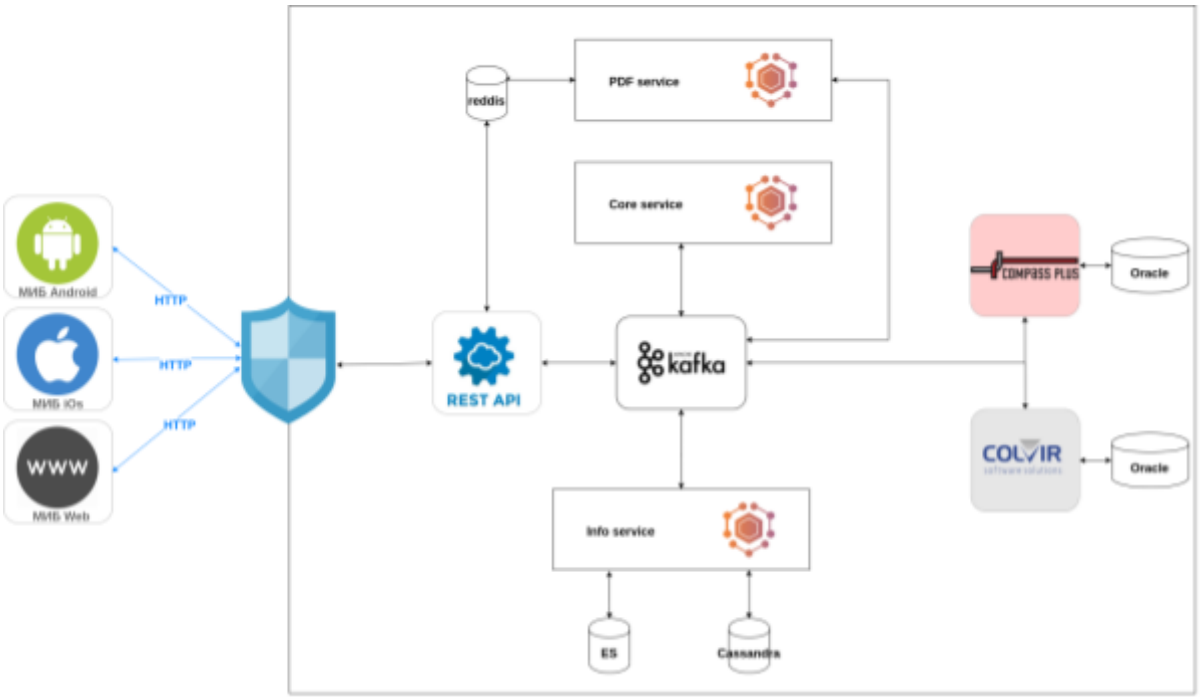

Рисунок 4.5 - Схема работы бизнес-процесса

На рисунке 4.5 изображена схема работы бизнес-процесса справок. При получении запроса API передает запрос в Core service. Далее необходимо получить информацию, поэтому производится запрос в Info service, который в свою очередь запрашивает информацию из АБС Colvir и Compass Plus. После получения информации, она сохраняется в базах данных ElasticSearch и Cassandra, для того, чтобы при считывании QR-кода из справки, была сформирована та же самая справка. На следующем шаге, вся собранная информация отправляется в PDF service, где формируется уже сам файл справки в том виде, в котором получает пользователь и кодируется в формат Base64, а также записывается в виде кэша в Redis с уникальным идентификатором. Далее ответ о формировании возвращается в API, где по уникальному идентификатору считывается с Redis и отправится по HTTP, как ответ на запрос. Все взаимодействие между сервисами происходит с помощью брокера сообщений Apache kafka.

4.2 Пути развития

В ходе будущего развития и улучшения по проекту будет усовершенствование имеющихся процессов или полная переработка некоторых из них, в связи с обратной связью пользователей.

Также планируется работы по масштабированию всей системы, так как, в связи с эпидемиологической ситуацией, количество клиентов интернет-банкинга возрастает с каждым днем. Возможно, что некоторые используемые технологии будут заменены на еще более новые и усовершенствованные. Но в ближайшее время такого не планируется в связи с тем, что новые технологии необходимо тщательно протестировать и изучить все аспекты использования, так как “молодое” программное обеспечение может иметь различные ошибки или недоработки из-за малого количества клиентов и сообщества.

Не исключен вариант полного отказа от сторонних разработок в пользу собственных. К таким разработкам может относиться Netflix Conductor и другие. Такое решение позволит реализовать технологию, четко заточенную под нужды интернет-банкинга, полностью подходящую под требования информационной системы, увеличит “отзывчивость” и уменьшит риски возникновения ошибок.

ЗАКЛЮЧЕНИЕ

В ходе разработки был проведен анализ финансового сектора, а именно, банковских систем в сфере информационных технологий и разработана информационная система интернет-банкинга, предоставляющая пользователям доступ к дистанционному банковскому обслуживанию.

Также был проведен анализ современных технологий, используемых в разработке, таких как:

- Netflix Conductor
- Акторной системы на основе Akka
- Баз данных на основе ElasticSearch и Cassandra

Все перечисленные технологии использовались при реализации дипломной работы.

Были решены следующие поставленные задачи:

- Проанализирован рынок автоматизированных банковских систем;
- Исследованы современные технологии разработки информационных систем;
- Разработана архитектура ИС с использованием изученных технологий;
- Разработанная система была проанализирована и протестирована. Поставлены задачи будущего развития.

В итоге необходимо отметить и вышеуказанные пути развития информационной системы интернет-банкинга, которые включают в себя масштабирование и улучшение работы всей системы в целом, а также разработку собственных, узкоспециализированных технологий.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Akka Actors [Электронный ресурс]. - https://doc.akka.io/docs/akka/current/typed/index.html?_ga=2.170799849.819075291.1621098211-1276339580.1601890559
2. Apache Kafka Documentation [Электронный ресурс]. - https://kafka.apache.org/documentation/
3. Colvir [Электронный ресурс]. - https://www.colvir.com/
4. Compass Plus [Электронный ресурс]. - https://compassplus.ru/
5. Netflix Conductor [Электронный ресурс]. - https://netflix.github.io/conductor/
6. Scala Documentation [Электронный ресурс]. - https://docs.scala-lang.org/?_ga=2.88166488.293595442.1621098378-593662773.1601192873
7. Введение в REST API [Электронный ресурс]. - https://habr.com/ru/post/483202/
8. Микросервисы (Microservices) [Электронный ресурс]. - https://habr.com/ru/post/249183/
9. Redis Documentation [Электронный ресурс]. - https://redis.io/documentation
10. Cassandra Documentation [Электронный ресурс]. - https://cassandra.apache.org/doc/latest/
11. ElasticSearch Documentation [Электронный ресурс]. - https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
12. Введение в sbt [Электронный ресурс]. - https://habr.com/ru/post/231971/
13. Автоматизированные банковские системы (АБС) [Электронный ресурс]. - http://eos.ibi.spb.ru/umk/5_12_11/5/5_R1_T2.html
14. Kafka Documentation [Электронный ресурс]. - https://kafka.apache.org/documentation/

ПРИЛОЖЕНИЕ А

```
class StatementActor(publisher: ActorRef, redis: Redis)(implicit
    override val endpoint: Endpoint,
    system: ActorSystem,
    val config: Config,
    val timeout: Timeout,
    executionContext: ExecutionContext,
    val logSystemName: String,
) extends LoggingActor
    with SerializersWithTypeHints
    with CoreService {
  def extractFutureResponse[T: Typeable](future: Future[_]): Either[T, Any] = {
    val typeCase = TypeCase[T]

    Await.result(
      future,
      Duration(15, TimeUnit.SECONDS),
    ) match {
      case typeCase(r) => Left(r)
      case any => Right(any)
    }
  }
```

```
override def receive: Receive = {
  case request: GetStatementOfAccountPDFRequest =>
    stan = getStan(headers = request.headers)
    log.info("Received Request = {}", request)
    replyTo = sender()

    log.info(s"... waiting statementsResults")
    val statementsResultsFutures = request.body.languages.map { lang =>
      getStatementOfAccountFuture(
        request.body.profileId,
        request.body.iban,
        request.body.accountType,
        request.body.athensClientId,
        request.body.isShowBalance,
        lang,
        request.userContext,
      )
    }
    val statementsResults = statementsResultsFutures.map(extractFutureResponse[GetStatementOfAccountResponseBody](_))
    log.info(s"statementsResults = $statementsResults")
```

```
      val errorResultStatements = statementsResults.find(p => p.isRight)
      if (errorResultStatements.isDefined)
       errorComplete(request, errorResultStatements.get.right.get)
      else {
       log.info(s"... waiting createQRRegistryResults")
       val createQRRegistryResultsFutures =
        statementsResults.map(_.left.get.statement).map { statement =>
         createQRRegistryQRFuture(statement, request.userContext)
        }
       val createQRRegistryResults =

createQRRegistryResultsFutures.map(extractFutureResponse[CreateQRRegistryR
esultBody](_))
       log.info(s"createQRRegistryResults = $createQRRegistryResults")
       val errorQRRegistry = createQRRegistryResults.find(p => p.isRight)
       if (errorQRRegistry.isDefined)
        errorComplete(request, errorQRRegistry.get.right.get)
       else {
        val qrDatas = createQRRegistryResults.map(_.left.get.qrData)
        log.info(s"... waiting getQRStatementPDFResult")
        val getQRStatementPDFResult =
         extractFutureResponse[GetQRStatementPDFResponseBody](
          getQRStatementPDFFuture(qrDatas, request.userContext),
```

```
            )


          if (getQRStatementPDFResult.isRight)
            errorComplete(request, getQRStatementPDFResult.right.get)
          else {
            log.info(s"success getQRStatementPDFResult")
            extractPdf(
              getQRStatementPDFResult.left.get.pdfKey,
              replyTo,
                pdf => replyTo ! GetStatementOfAbsenceOfDebtOnLoansPDFApiResponse(pdf = pdf),
            )
          }
        }
      }
    }

private final def extractPdf(pdfKey: String, errorTo: ActorRef, onPdf: String => Unit): Unit =
      redis.get(pdfKey).onComplete {
        case Success(pdf) => onPdf(pdf.get)
        case Failure(_) => errorTo ! ServerErrorRequestException
      }
    }
```

ПРИЛОЖЕНИЕ В

```
class StatementCoreProxy(kafkaSinkActor: ActorRef)
                        (implicit config: Config,
                         endpoint: Endpoint)
 extends BaseProxy {

 override def unhandled(message: Any): Unit = message match {
  case _ =>
    log.info("Received unhandled command message = {}", message.toString)
    super.unhandled(message)
 }

 override def receive: Receive = {
  case command: StatementCoreProxy.Command => {
    val senderRef = sender()
    context.become(waitingForResponse(senderRef))
    kafkaSinkActor ! KafkaProducer.makeRecord(
      Map.empty,
      command.message,
      command.body,
      Some(endpoint.instanceEndpoint),
      Some(actorName),
```

```
        command.userContext
      )
    }

    case request: StatementCoreProxy.Request => {
      val senderRef = sender()
      context.become(waitingForResponse(senderRef))
      kafkaSinkActor ! KafkaProducer.makeRecord(
        Map.empty,
        request.message,
        request.body,
        Some(endpoint.instanceEndpoint),
        Some(actorName),
        request.userContext
      )
    }
  }
  def waitingForResponse(senderRef: ActorRef): Receive = {
    case any =>
      log.info(s"Received response/result: $any")
      senderRef ! any
  }
}
```