Санкт-Петербургский государственный университет

Шапошников Алексей Игоревич

Выпускная квалификационная работа

Онбординг бизнес-потребителей в системе финансового процессинга

Уровень образования: бакалавриат

Направление *09.03.04 «Программная инженерия»*

Основная образовательная программа *СВ.5080.2018 «Программная инженерия»*

Научный руководитель:
к.ф.-м.н., доцент Д.В. Луцив

Рецензент:
генеральный директор ООО «ВИНГУ» А.В. Суслов

Санкт-Петербург
2022

Saint Petersburg State University

Shaposhnikov Alexey Igorevich

Bachelor's Thesis

Onboarding of business consumers in the financial processing system

Education level: bachelor

Speciality *09.03.04 «Software Engineering»*

Programme *CB.5080.2018 «Software Engineering»*

Scientific supervisor:
C.Sc., docent D.V. Luciv

Reviewer:
general director at "Veengu" A.V. Suslov

Saint Petersburg
2022

Оглавление

Введение

Одной из главных задач любого бизнеса, предоставляющего программные продукты или Интернет-сервисы, является удержание клиентов. Как бы не был хорош, удобен в использовании сервис, как бы он не превосходил конкурентов, это не будет иметь значения, если потенциальный пользователь или клиент не пройдет дальше этапа регистрации или погружения. Поэтому перед командами разработчиков стоит задача не только создать качественный, конкурентоспособный программный продукт, но и обеспечить удобный и легкий процесс ознакомления клиентов с продуктом.

С первого взгляда кажется, что подобная задача не имеет отношения к бэкенд-части систем и зависит только от проработанного визуального интерфейса, но это не всегда так. Например, если обратиться в финансовую и банковскую сферу, то начало взаимоотношений с клиентом не ограничиваются простой регистрацией и подтверждением контактных данных - специальным термином onboarding (далее онбординг) обозначается прохождение всех этапов для получения финансовой услуги.

Онбординг в банковских сервисах может различаться в зависимости от видов клиентов (например, физическое или юридическое лица) и цели обращения. Так, онбординг может включать такие этапы, как указание персональных данных (например, паспортных данных), приложение документов (например, сертификата о налоговом резидентстве), выбор тарифных планов, создание и подписание контракта, а также оплата услуг.

Чем больше таких этапов содержит сервис до получения непосредственно самой услуги, тем сильнее возрастает важность прозрачности этих этапов и удобства их прохождения. Поэтому становятся востребованными специальные системы для реализации этой функциональности.

В области информационных систем термином “белая этикетка” (white label) принято называть продукт или сервис, который может использо-

ваться различными клиентами без дополнительных затрат на создание технологий и инфраструктуры. Например, мелкие банки могут использовать кредитные карты большого банка; клиенты могут взять готовое веб-приложение и сайт и внести косметические/брендовые изменения. Тогда клиенты продуктов с “белой этикеткой” вводят свои данные в готовое решение и могут даже менять его рабочий процесс. С точки зрения бизнеса разработка такого решения позволяет увеличить количество клиентов, поставляя услугу не только напрямую (b2c [3], business-to-consumer), но и через дистрибьюторов (b2b [2], business-to-business). И если создать систему онбординга с ”белой этикеткой”, то банк сможет продавать и само решение, и финансовые продукты другим бизнесам, которые под своим брендом будут доставлять эти продукты конечным пользователям.

У одной финансовой организации, предоставляющей компаниям финансовые решения для реализации различных платежей, как раз появилась необходимость в такой системе, позволяющей настраивать схему рабочего процесса, каталог продуктов, при необходимости расширять и добавлять новые этапы и интеграции. Планируемая система должна позволить разбивать сложный процесс онбординга для бизнес-потребителей на понятные этапы и Мастер-шаги, а также быть повторно используемой для различных дистрибьюторов, чтобы увеличить охват рынка.

Постановка задачи

Целью данной дипломной работы является проектирование и реализация настраиваемой системы для онбординга бизнес-клиентов в банковской сфере. Для достижения этой цели в рамках данной работы были сформулированы следующие задачи.

- Произвести обзор существующих решений в области онбординга;
- Разработать функциональные требования к новой системе онбординга для предоставления финансовых решений;
- Спроектировать архитектуру:
  - разработать схемы этапов онбординга и плана продуктов;
  - спроектировать взаимодействие с слоем данных;
- Реализовать решение:
  - реализовать расширение Keycloak;
  - реализовать интеграции в виде микросервисов;
  - поддержать различные роли пользователей;
- Провести тестирование и апробацию получившегося продукта;
- Внедрить разработанную систему в промышленное использование.

Существующие решения и технологии

Сама по себе разработка удобной системы для онбординга или wizard [28] (”волшебный помощник” или Мастер) далеко не нова и уже не представляет инженерной ценности, поскольку может быть осуществлена множеством способов, инструментов или даже low-code технологий. Однако же задача, поставленная в данной выпускной работе, осложнена конкретными бизнес-требованиями и концепциями. Поэтому обратимся к тому, как происходит онбординг клиентов в различных предметных областях, и рассмотрим косвенные аналоги.

Обзор существующих решений

Регистрация в нетфликс

Со стороны развлекательных сервисов можно рассмотреть Netflix [16], онбординг в котором состоит из следующих этапов:

1. Начальная авторизация пользователя по почте.
2. В случае, если аккаунт с данной почтой не зарегистрирован, система предлагает придумать пароль, приободряет словами ”We hate paperwork, too” и указывает, сколько еще шагов осталось пройти.
3. Затем идет выбор плана подписки из предложенных, в которым они сравниваются по различным критериям (как качество видео, разрешение, цена).
4. Указание способа оплаты и реквизитов счета.
5. Выбор трех медиа ресурсов из предложенных, по которым будет строиться начальная рекомендательная система.

Несмотря на то, что в рассмотренной регистрации всего три простых этапа, компания задумалось о пользовательском опыте: нет кнопок авторизации или регистрации, нужно лишь ввести почту на главном экране, указывают оставшееся количество шагов и воодушевляют

пользователей продолжить процесс регистрации. Такой подход позволяет терять меньше клиентов, которые решили пройти регистрацию, но потеряли интерес или устали от заполнения данных посреди процесса и закрыли окно.

Онбординг юридических лиц в Тинькофф банке

Рассмотрим также онбординг со стороны финансовых сервисов для корпоративных клиентов, что более близко к нашей предметной области и подразумевает более сложный процесс регистрации. Возьмем для рассмотрения Тинькофф банк, поскольку он находится в банковском секторе, при этом ориентирован на простой и приятный опыт использования.

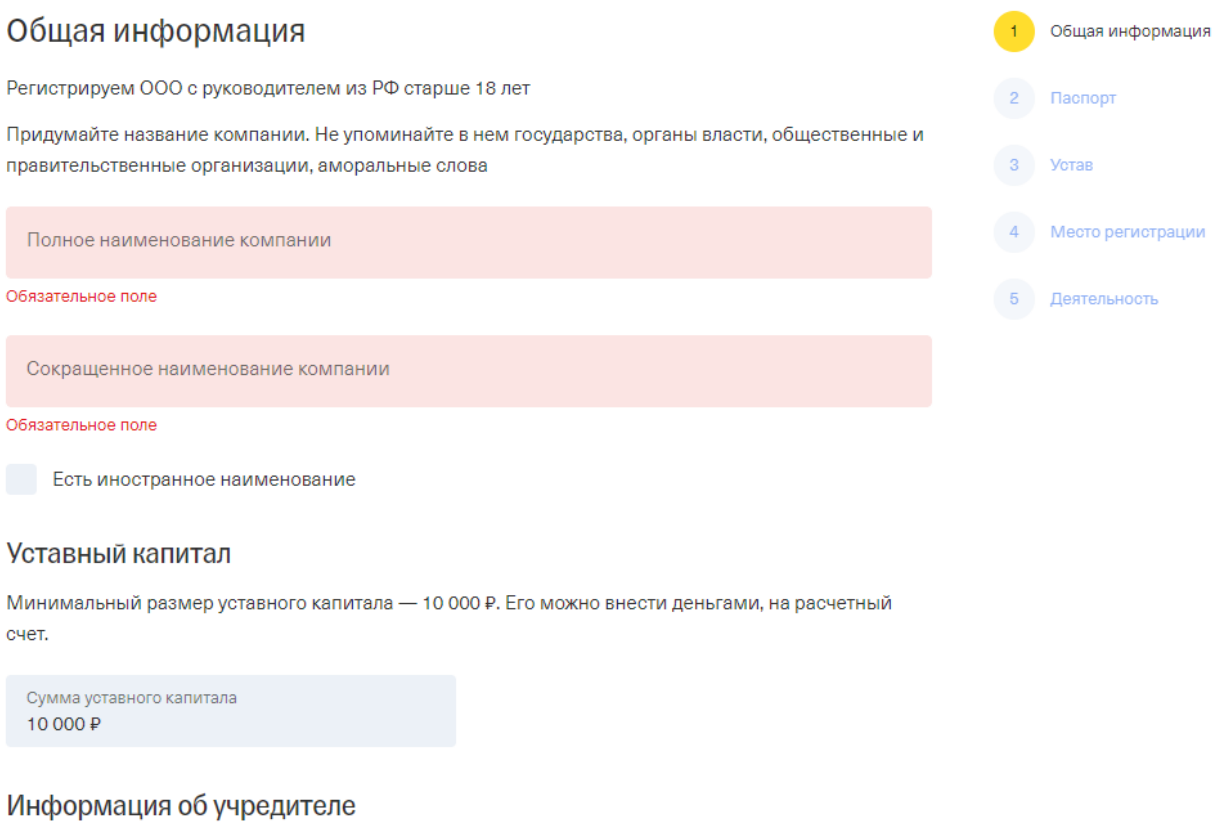

Рис. 1: Регистрация юр. лица в Тинькофф

Процесс регистрации юридического лица [31] проходит так.

- После подтверждения телефона или входа в уже существующий

аккаунт, начинается онбординг для регистрации ООО;

- Как видно на Рис. 1, мы стоим на 1 из 5 шагов по заполнению заявки без возможности перейти на следующий шаг, пока не заполним текущий. Шаги разделены по предметной области, каждый подразумевает заполнение данных о юридическом лице, предприятии, деятельности и т.д. Данный этап сбора данных также подразумевает загрузку документов с использованием шаблонов и подсказок от банка;

- По окончанию формирования заявки, вся информация и прикрепленные документы будут проверены и отправлены в налоговую;

- При успешной проверке заключительный этап – личная встреча с представителем банка и подписание.

Рассмотренный процесс близок к поставленной бизнес-задаче по следующим критериям.

- Он содержит как внутренние шаги (этапы заполнения заявки), так и внешние (этапы проверки и подписания документов);

- Онбординг содержит шаблоны для загрузки документов, подсказки, проверку документов и уведомления о результатах по почте;

- Кол-во и информация шагов в заполнении заявки меняется в зависимости от типа юридического лица (ИП или ООО).

Гибкая конфигурация в Salesforce

Теперь рассмотрим косвенный аналог, который интересен нам не подходом к регистрации клиентов и их онбордингу, а в предоставляемой функциональности гибко конфигурировать свои рабочие процессы, доменную область, не вынуждая глубоко погружаться в технологию, писать или править исходные коды, а пользоваться встроенными интерфейсами для low-code.

Salesforce [22] — CRM-платформа [4], используемая для управления контактами, продажами, бизнес-отношениями, взаимодействия с клиентами и оптимизации процессов. Однако обозревать решаемые этой платформой задачи и технологические решения – вне области данного обзора, поэтому обратим внимание на то, как они позволяют создавать пользовательские конфигурации:

- Salesforce предоставляет возможность настроить слой данных путем создания необходимых сущностей через экраны панели администрирования. Фактическое создание таблиц, хранение и управление данными платформа берет на себя;

- как показано на Рис. 2, администраторы имеют возможность создавать схемы процессов, добавляя в них необходимую логику обработки событий, проверки данных, времени запуска, отправки уведомлений и прочее;

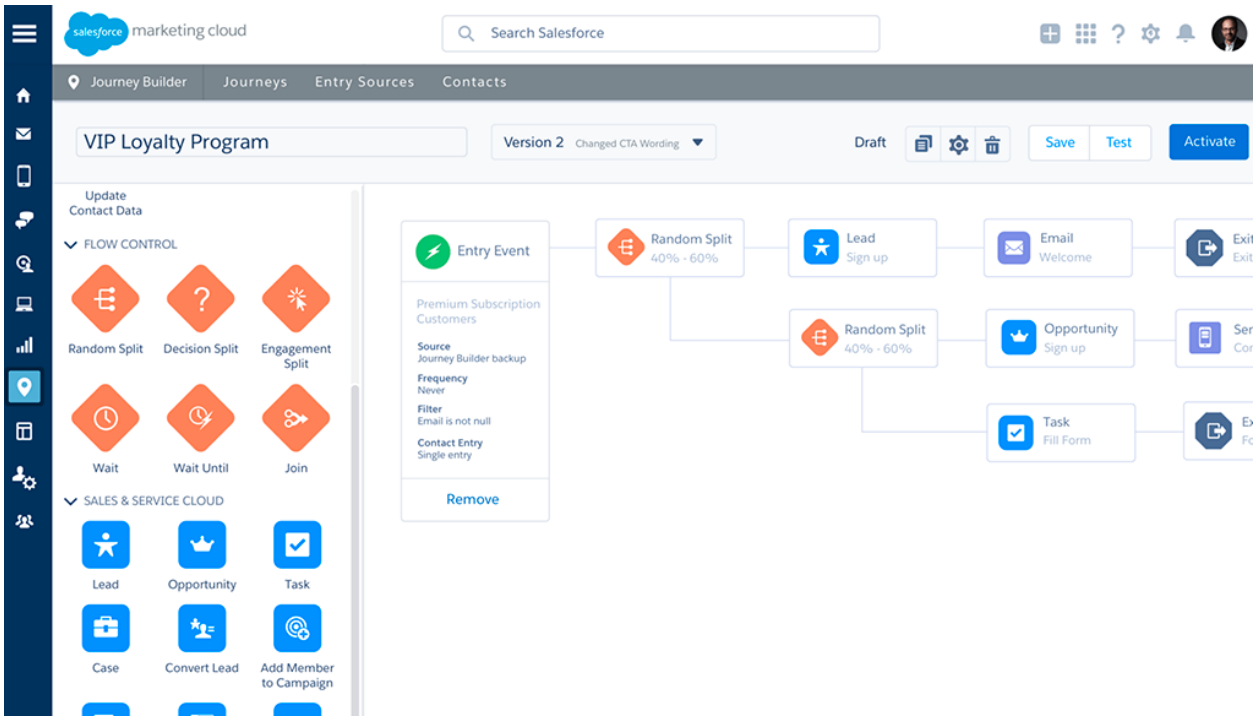

Рис. 2: Пример схемы процессов Salesforce

- для добавления своей функциональности, которая не покрывается предусмотренными базовыми сценариями, или создания но-

вых визуальных интерфейсов, платформу предоставляем возможность расширения через язык программирования Apex [1] или язык разметки Visualforce [27].

Вывод

Рассмотренные аналоги по различным аспектам и возможностям пересекается с определенным нами доменом. Обобщим обзор, определив плюсы и минусы, а точнее поддержку возможностей у этих технологий.

Таблица 1: Сравнение косвенных аналогов

Критерий\решение	Регистрация Netflix	Онбординг юр. лиц в Тинькофф	Административная панель Salesforce
Простая аутентификация	+	−	−
Разбиение онбординга на шаги	+	+	n/a
Разбиение онбординга на этапы	−	+	n/a
Полностью online решение	+	−	+
Поддержка разных ролей	неизвестно	неизвестно	+
Поддержка no-code или low-code изменения конфигурации	−	−	+
Возможность перехода между Мастер-шагами без отправки данных с сохранением состояний	−	−	n/a
Возможность работы с одной заявкой различных пользователей и агентов	−	−	n/a

Приведенная таблица 1 интересна нам не столько для сравнения конкурентов и принятия решения о проектировании продукта, который учтет недостатки, сколько для проведения аналогий с другими технологиями по интересующим нас возможностям.

Как видно, искомыми характеристиками обладают те или иные решения в зависимости от области их применения, но не какой-то продукт всеми сразу: это связано с тем, что перечисленные аналоги хорошо решают поставленную перед ними задачу и предоставляют для этого необходимые пользовательские или программные интерфейсы, а в таких возможностях, как поддержка различных ролей, конфигурация этапов онбординга и т.п., они просто не нуждаются.

Поэтому есть определенный интерес спроектировать и реализовать решение, которое поддержит возможности технологий из разных обла-

стей для гибкого онбординга бизнес-клиентов с поддержкой концепции "белая этикетка".

Использованные технологии

Spring Boot

Spring Boot [8] — одна из самых популярных платформ для создания решений и платформ корпоративного масштаба для JVM, которая предоставляет высокий уровень абстракции – благодаря чему можно как создавать новые микросервисы минимальными усилиями, так и глубоко настраивать конфигурацию сервиса, интеграции с технологиями и т.д. От Spring Framework [18] платформа отличается меньшими требованиями к созданию сервиса: Spring Boot поставляет большинство необходимых классов с стандартной конфигурацией уже в контейнере зависимостей (с возможностью переопределения) и различные "стартеры" (например, для web, валидации, работы с базами данных, логгирования и т.д.), которые загружают все необходимые зависимости и автоконфигурируют их.

Так как суть работы в реализации кастомизируемой системы, которая будет переиспользоваться различными бизнес-клиентами/тенантами, то было принято писать на широко известном и поддерживаемом JVM языке, Java. При этом одним из требований со стороны заказчика стало использование версии Java 8 [10]: уже достаточно устаревшей, если сравнивать с новыми языками программирования и версиями Java (на момент написания работы уже выпущена 17 версия [29]), но все еще встречающейся в legacy-проектах и поддерживаемой большим количеством проектов благодаря обратной совместимости [30]).

Возвращаясь к Spring, отметим преимущества, которые помогли принять решение о выборе технологии.

- Удобное управление зависимостями и жизненным циклом объектов в контейнере;
- Простая интеграция с известными технологиями;

- Широкая распространенность и поддержка;
- Широкие возможности создания Web-приложений и облачных сервисов;
- Гибкая система управления транзакциями.

Hibernate

Когда речь заходит о Spring приложениях и Java, вместе с ними сразу вспоминается Hibernate [19] — популярная ORM-технология [17], работающая над JDBC [21] и реализующая спецификацию JPA [9]. Она обеспечивает связь с слоем данных, позволяя обращаться к различным реляционным СУБД, в которых неизбежно возникает необходимость при разработке корпоративных приложений.

Spring обеспечивает хорошую поддержку спецификации JPA и транзакционным механизмам и имеет высокую абстракцию над уровнем данных, поэтому эта комбинация очень популярна.

В данной работе будет использоваться MySQL, согласно поставленным требованиям, поэтому в анализе того, какая СУБД лучше подойдет под поставленную задачу, не было необходимости.

Keycloak

Keycloak [12] — сервис управления аутентификацией, который "из коробки" позволяет идентифицировать пользователей различными способами (через телефон и почту, через логин и пароль, OTP и т.д.) и их комбинациями. Keycloak берет на себя ответственность за хранение данных и предоставляет административный доступ к его настройке, управлением пользователями, realm [20] и т.д.

Keycloak не только дает выбор для использования конкретного способа аутентификации из поддерживаемых, но и позволяет создавать индивидуальный сценарий с помощью расширения [14] JEE [7] приложения.

Анализ требований

В ходе обсуждения бизнес-требований были сформулированы следующие функциональные требования.

- Упрощенная аутентификация: пользователь вводит либо телефон, либо почту, получает OTP код (One Time Password), вводит и либо начинает процесс онбординга, либо продолжает с последнего незаконченного шага;
- Выбор коммерческого продукта или плана:
  - на серверной части хранится конфигурационный файл со всеми данными и метаданными, согласно которому предлагаются различные планы и комбинации, показываются дополнительные формы для заполнения, идет подсчет цены;
  - должны быть поддержаны скидки и индивидуальные условия, которые предоставляются агентом отдельному клиенту;
- Схема данных и шаги не связаны с логикой серверной части или слоя данных:
  - для каждого шага есть валидация;
  - для каждого шага есть актуальное состояние других шагов, на которые можно перейти, которые недоступны и т.п.;
  - не подтвержденные данные шага (черновики) так же сохраняются и видны разным клиентам одного заявления;
  - все заполненные пользователем данные хранятся для тенанта (клиента), который предоставляет процедуру онбординга;
- Этапы или стадии онбординга предполагают:
  - формирование и подписание контракта;
  - интеграцию с платежными сервисами;

  - интеграцию с зависящими от тенанта сервисами: хранилище документов, проверка указанных физических и юридических лиц, отправка заявления в CRM;
  - поддержку различных состояний (в том числе терминальных) по результатам различных процессов, как проверка контактных лиц, валидация данных в CRM и прочие;

- Поддержка различных ролей для суперпользователей (агент и руководитель команды) с аутентификацией через Keycloak; дополнительная функциональность агентов:
  - изменение и отправка данных заявления пользователя;
  - удаление заявки либо продвижение по статусу (например, "рассмотрение одобрено");
  - изменение цен договора в индивидуальных случаях;

- Следующие элементы онбординга должны быть конфигурируемы без изменения исходного кода для поддержки концепции "белая этикетка":
  - ценообразование;
  - добавление новых продуктов или планов;
  - заполняемые поля и шаги.

Архитектура

Если взять во внимание специфику области создаваемой системы, функциональные требования и необходимость поддержать концепцию ”белая этикетка”, становится ясно, что проектирование архитектуры — существенная часть в реализации решения, и важно сразу учитывать ограничения, которые это накладывает на систему.

Взаимодействие модулей системы

Для взаимодействия между модулями системы и внешними интеграциями была спроектирована диаграмма контейнеров, указанная на Рис. 3.

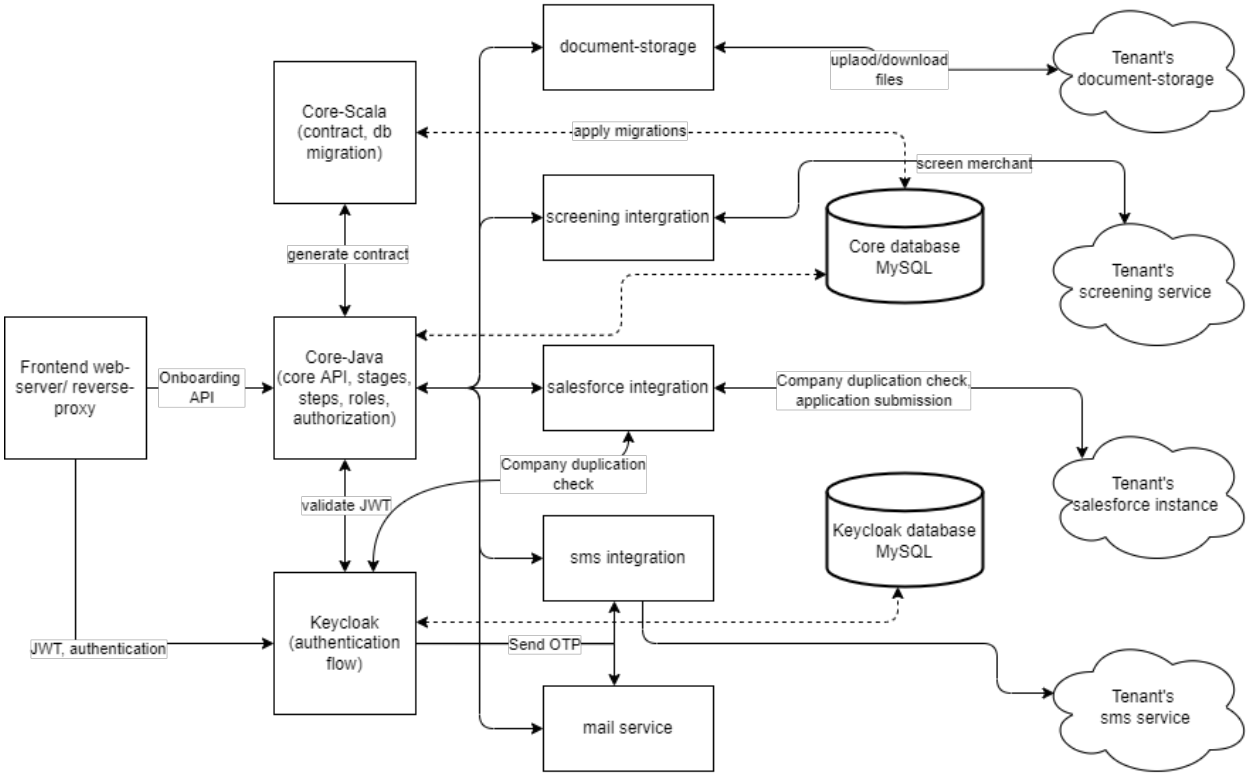

Рис. 3: Диаграмма контейнеров

- Точка входа в систему для клиента – фронтенд веб-приложения, от которого запросы направляются на обратный прокси-сервер [25];
  - запросы к сервису аутентификации Keycloak позволяют пройти пользователю аутентифицироватсья и получить JWT [11]

(JSON Web Token) для дальнейшей авторизации в системе, в котором содержится вся необходимая информация;

  - ‘Core-Java‘ сервис предоставляет клиенту весь API для прохождения онбординга, смену Мастер-шагов и этапов, функциональность супер-пользователей. Запросы к кор сервису должны содержать заголовок текущего тенанта/клиента (название компании) и токен аутентификации;

- Основной сервис, написанный на Java, обращается к инстансу Keycloak для валидации токена, к взятому из существовавшей кодовой базы Scala-сервису для генерации контракта, а также к сервисам интеграции:

  - и Keycloak, и Java-сервис обращаются к смс и почтовым сервисам для отправки информации об OTP-кодах и уведомлениях о событиях онбординга соответственно. Смс сервис является интеграцией к конкретному стороннему сервису клиента, как и остальные сервисы-интеграции;

  - к главной базе данных применяются миграции при запуске Scala-сервиса, а затем она активно взаимодействует с основным Java-сервисом;

  - для загрузки и скачивания файлов используется интеграция с хранилищем данных клиента;

  - для процесса скрининга (отсмотра нежелательных физических лиц) используется интеграция с сторонним screening-сервисом;

  - также существует интеграция с CRM-системой Salesforce для запросов проверки дупликации (одна компания может проходить онбординг только один раз) и отправки заявки со всеми данными в систему хранения и обработки данных клиента;

- Java-сервис, Scala-сервис, Keycloak-сервер с его отдельной базой данных, основная база данных на MySQL, сервер с статически-

ми UI ресурсами, а также сервисы интеграции находятся в одном кластере и взаимодействуют друг с другом через обратный прокси-сервер;

- Сторонние сервисы, предоставляемые клиентом, находятся вне кластера – для них указываются актуальные реквизиты в сервисах интеграции.

Такая схема взаимодействия с внешними сервсиами позволяет вынести из главного Java-сервиса килент-специфический API в отдельные сервисы интеграции, которые при необходимости могут заменяться клиентом при разворачивании его собственной инфраструктуры.

Также предусмотрен вариант, при котором клиент будет использовать свой кастомизированный UI и делегировать запросы кластеру со стороны заказчика.

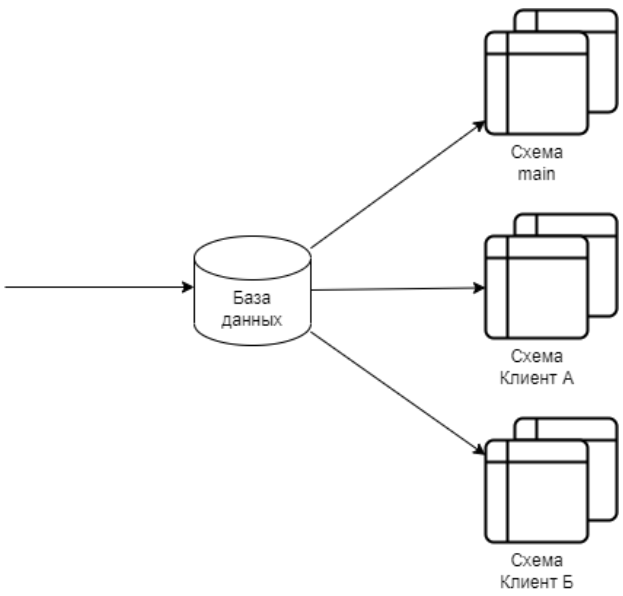

Рис. 4: Мультиарендная база данных

Чтобы поддержать возможность использовать один стенд для разных клиентов (тенантов), не внося изменений в исходный код, были предприняты следующие дизайн-решения о взаимодействии с уровнем данных.

- Поддержка концепции мультиарендности (multitenancy [5]), как показано на Рис. 4: она подразумевает, что в одной базе данных может находиться несколько дубликатов главной схемы (со всеми нужными таблицами) для каждого зарегистрированного клиента. Задача регистрации тенанта, создания для него отдельной схемы и поддержки акутального состояния всех миграций была решена в Scala-сервисе;

- Все запросы, во время исполнения которых происходит обращение к базе данных, должны содержать информацию о текущем тенанте. Поэтому веб-интерфейс, предоставляемый конкретным тенантом, во всех запросах содержит заголовок с этой информацией.

Конфигурация Мастер-шагов онбординга

Один из самых объемных и важных этапов для онбординга юридического лица – сбор всех необходимых данных и документов. И, согласно составленным требованиям, Мастер-шаги и поля должны быть настраиваемыми без вмешательства в исходный код.

Поэтому было решено, что серверная часть изначально не будет знать о данных конкретной заявки – она будет храниться в формате json в базе данных. А задачи определения списка шагов, их видимости для пользователя и валидации отправляемых данных будут возложены на внешние ресурсы (json-схемы и JavaScript файлы), которые можно заменять без изменений исходного кода или необходимости его перекомпилировать.

При таком подходе json-схема как конфигурация Мастер-шагов должна содержать для каждого шага следующую информацию.

- Полный ключ и ключ для отображения на UI;

- Название функции для разрешения видимости. Например, какие-то шаги видны всегда, для каких-то нужны определенные условия;

- Префикс пути для отправляемых в виде json данных. Это нужно, чтобы, находясь на одном шаге, нельзя было повлиять на уже сохраненные данные других шагов, т.к. можно отправить только значения с указанными префиксами в их ключах;

- Шаги, от которых зависит текущий. Эта информация может понадобиться, если есть логически связанные шаги – так, что если информация в одном поменялась, то в другом могут появиться или исчезнуть поля, а его состояниее должно стать "не отправлено";

- Список под-шагов – для удобства конфигурирования.

Схема продуктов

Если набор необходимых данных заявки – относительно известная, редко меняющаяся вещь, то предоставляемые услуги или продукты, тарифные планы – предмет частых изменений. Также стоит отметить, что клиент при адаптации решения по онбордингу с "белой этикеткой" должен иметь возможность предложить свои планы, поменять конкретные условия, не предоставлять какой-то продукт.

Из этих условий возникает необходимость в конфигурируемой схеме продуктов – то есть такой, что она будет содержать необходимый и достаточный объем информации в том числе для отображения на клиентской части (потенциально изменения в схему могут вноситься ежедневно, поэтому клиентская часть должна полагаться полностью на схему). Путем множества итераций по требованиям, предоставленному набору данных и возможным взаимодействиям с продуктами, была спроектирована схема, содержащая список продуктов со следующей информацией.

- Тип продукта (из множества известных). Например, решение по электронной коммерции;

- Ключ, название, ссылка на статический ресурс с картинкой продукта;
- Описание, заголовок, ссылка на помощник по данному продукту;
- Дополнительная информация. Например, для электронной коммерции здесь перечислены способы веб-интеграции;
- Список полей продукта – где поля представляют информацию о цене, тарифах, комиссиях, подключенных услугах и поддерживаемых платежных схемах:
  - тип, по которому определяется, как это поле будет отображено клиенту;
  - ключ и название;
  - секция или категория, к которой поле относится (для клиентского отображения);
  - влияет ли на первичную цену приобретения услуги (для подсчета стоимости);
  - подсказка и описание при необходимости;
  - видно ли это поле пользователю. Если не видно, то оно не показывается, но учитывается в контракте и CRM-заявке;
  - значение поля – в простом виде может быть строкой или логическим типом данных, но для ценовых полей указывает либо проценты, либо сумму и валюту, а также примененные скидки.

Особенности реализации

Однако же проектирование без соответствующей ему реализации остается благонамеренным мыслительным процессом.

Поддержка конфигурируемых этапов онбординга

Возвращаясь к схеме Мастер-шагов, описанной в предыдущей главе, необходимо отметить, как реализовано взаимодействие с ней на серверной части и как проходит прохождение этапа заполнения данных.

В качестве внешних ресурсов также используются JavaScript файлы и необходимые для их функционирования зависимости. Например, библиотека для работы с датами или валидации телефонного номера.

- Файл валидации шагов на JavaScript – валидирует заполненные пользователем данные в соответствие с ключом текущего шага;
- Файл определения видимости шагов на JavaScript – исходя из уже известных данных и последнего отправленного шага, решает, какие шаги видны пользователю и на какой следующий шаг он попадает. Например, если пользователь указывает, что тип собственности компании – индивидуальный предприниматель, то шаг с данными акционеров скрывается.

В сущности пользователя запоминаются уже пройденные Мастер-шаги и последний актуальный – поэтому при входе в веб-приложение он попадает на свой последний шаг.

При выборе шага учитывается схема доступных конкретному пользователю шагов.

Отправленные пользователем данные не типизируются и представляют собой json, который (при успешном прохождении валидации) сливается с основным json-ом, содержащим все данные заявки и хранящимся в базе данных. Если у пользователя успешно пройдены все Мастер-

шаги, то после отправки данных последнего шага происходит автоматическая смена этапа.

Поддержка концепции мультиарендности в модели данных

Hibernate предоставляет различные опции для реализации концепции мультиарендности во взаимодействии с базами данных. Для этого надо в настройках JPA [9] (Jakarta Persistence API, спецификация взаимодействия с данными, которую реализует Hibernate) задать ключу *hibernate.multitenancy* способ реализации концепции мультиарендности (в рассматриваемом случае – на основе схем) и зарегистрировать провайдер мультиарендных соединений (*MultiTenantConnectionProvider*) и разрешатель идентификатора текущего тенанта (*CurrentTenantIdentifierResolver*).

Разрешатель идентификатора тенанта обращается к контексту текущей сессии и возвращает переданный заголовок с названием компании. Провайдер мультиарендных соединений по переданному идентификатору нативным запросом в MySQL меняет схему текущего (случайно взятого) соединения. При выпуске соединения используемая схема заменяется на главную.

Интеграция микросервисов

Сервисы интеграции реализованы в качестве Spring веб-приложений, предоставляют REST API и общаются по HTTP внутри кластера.

Большая их часть представляет собой прослойку к сторонним сервисам, но есть и сложные случаи, как интеграция с CRM-системой, которая требует своей логики отправления данных, правил валидации и проч.

Почтовый сервис может использовать библиотеку Spring для отправки почты [23] либо вызывать Sendgrid API [26] в зависимости от профиля.

Расширение аутентификации Keycloak

Keycloak предоставляет готовые сценарии для разных путей аутентификации. Однако поставленная задача предполагала собственные уникальные сценарии (например, вход по приглашению и подтверждение OTP по телефону), поэтому возникла необходимость в написании пользовательского расширения.

В сущности, расширение Keycloak представляет из себя проект на JEE [7], состоящих из классов, расширяющих существующие классы и интерфейсы Keycloak, создавая новую логику или дополняя существующие сценарии. Например, для создания нового сценария можно создать свой аутентификатор [15], создать для него фабрику и зарегистрировать ее в мета-информации расширения. Затем в панели администратора можно настроить сценарии аутентификации, различных клиентов, их роли и атрибуты, обязательные действия при аутентификации, какая информация попадает в JWT и многое другое.

В случае сервиса онбординга пришлось расширять логику для аутентификации через OTP (переиспользуя генерацию одноразового пароля выбранного в панели типа, сохранение данных, защиту от лобовой атаки), создавать отдельные RequiredAction для указания первоначальных данных пользователя. Готовую аутентификацию удалось использовать только для входа агентов по почте и паролю.

Для отображения не стандартных страниц аутентификации Keycloak, а нарисованных дизайнером, перед сборкой расширения в него подставляется индекс-файл, определяющий взаимодействие между клиентом и запросами/процессами Keycloak. После успешной аутентификации, клиент получает JWT с необходимыми данными пользователя и переходит на основной портал.

Поддержка роли суперпользователя

Разделение ролей пользователей осуществлено с помощью различных клиентов [13] в Keycloak, которые подразумевают отличющиеся сценарии аутентификации. По *azp* (authorized party) и *resource_access*,

передающимся в JWT, определяется природа текущего пользователя и его роли.

Если обычному пользователю всегда соответствует одна заявка, то агенту их доступно множество. Поэтому для прохождения онбординга в качестве обычного клиента в заголовок запросов дополнительно передается нужный идентификатор заявки. Информация о совершаемых агентом действиях сохраняется в ассоциативной сущности между агентом и завявкой.

Также суперпользователи имеют возможность менять любые данные заявки и возвращаться на предыдущие этапы. Поэтому написанная и отлаженная логика онбординга обычного пользователя потребовала рефакторинга, в том числе были переработаны все запросы, связанные с Мастер-шагами, и логика автоматического перехода на этап скрининга.

Другой функциональностью стала возможность агентов менять продуктовые планы, комиссии, выключать и включать платежные схемы. Схема продуктов так же подверглась рефакторингу, часть данных была выделена в виде отдельного Мастер-шага; к оставшимся полям были добавлены границы валидации и списки возможных значений.

Тестирование

Тестирование системы онбординга можно разделить на четыре части.

1. Юнит-тесты, подтверждающие, что код отдельных сервисов работоспособен на тестовых примерах.

2. Интеграционные тесты, ориентирующиеся на приближенные к реальным данным конфигурации и воссоздающие различные сценарии. Из-за большого количества внешних интеграций отдельные тесты покрывают конкретное ожидаемое поведение (а не полный пользовательский сценарий, например), а обращение к сторонним сервисам заменяется mock-результатами [24]. В таких тестах особое внимание уделено подготовке данных к тестированию.

3. Автоматизированные UI тесты, включающие полные пользовательские сценарии. Необходимость в таких тестах возникает из-за особенностей реализуемой платформы: код может работать, но связи с внешними схемами или сторонними интеграциями будут некорректными. Такие тесты запускаются на уже развернутом стенде, используя Selenium [6] – и могут также служить как регрессионое тестирование при сливании новых изменений. Созданием таких тестовых сценарием занимался отдельный сотрудник.

4. Ручное тестирование на стенде разработки нововведений согласно бизнес-требованиям.

Юнит-тесты и интеграционные тесты относятся к сиходному коду сервисов и запускаются перед каждым развертыванием на стенд (если тесты не проходят, развертывание отменяется). Для них используются фреймворки JUnit (стандартный фреймворк для тестирования на Java), Mockito (тестирование с использованием mock-объектов) и встроенная в память база данных H2 (чтобы для тестовых сецнариев создавалась реальная база данных, но не было зависимости тестов от внешнних узлов). Покрытие строк кода этими тестами состалвяет более 70%.

Автоматизированные UI тесты и ручное тестирование пользовательских историй служат проверкой перед отправкой новых изменений заказчику для приемочного тестирования. По результатам приемочного тестирования могут быть отправлены запросы на доработку, исправление ошибок, изменение требований – либо изменения будут приняты в следующий релиз на промышленный стенд.

Результаты

В ходе выполнения выпускной квалификационной работы были достигнуты следующие результаты.

- Произведен обзор косвенных аналогов задачи онбординга юридических лиц в развлекательной, банковской и софтовой сферах;
- Выявлены функциональные требования системы для аутентификации, функциональности различных ролей, поддержки концепции ”белая этикетка”;
- Спроектированы микросервисная архитектура решения и модель данных;
  - разработаны конфигурации для этапов онбординга и продуктовых планов с помощью json-схем;
  - спроектировано и поддержано взаимодействие с уровнем данных с учетом концепции мультиарендности;
- Реализована система по онбордингу;
  - реализовано расширение Keycloak для аутентификации через email-OTP, sms-OTP, приглашение по ссылке и введение данных компании;
  - внедрены микросервисы для отправки sms, email, прохождения скрининга, загрузки файлов, интеграция с Salesforce;
  - поддержаны роли агента и тимлида, переиспользующие и дополняющие рабочий процесс пользователя;
- Произведено приемочное тестирование пользовательских историй и исправлены дефекты и недоработки со стороны аутентификации, безопасности, бизнес-логики и пользовательского опыта;
- Продукт был внедрен в промышленное использование со стороны заказчика и “заонбордил” первых реальных клиентов.

Спроектированная архитектура выдержала множество запросов на изменение со стороны заказчика и приемочное тестирование пользователей. В дальнейшем планируется использовать реализованную систему под брендом других бизнесов и проверить соответствие дизайна концепции “белая этикетка”.

Список литературы

[1] Apex. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.salesforcetutorial.com/introduction-to-apex-programming/.

[2] Business-to-business. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://en.wikipedia.org/wiki/Business-to-business.

[3] Business-to-consumer. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://en.wikipedia.org/wiki/Direct-to-consumer.

[4] CRM. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://en.wikipedia.org/wiki/Customer_relationship_management.

[5] An Efficient Schema Shared Approach for Cloud Based Multitenant Database with Authentication and Authorization Framework / Sanjeev Pippal, Vishu Sharma, Shakti Mishra, D.S. Kushwaha // 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. — 2011. — P. 213–218.

[6] Gundecha U., Avasarala S. Selenium WebDriver 3 Practical Guide. — Packt Publishing, 2018. — ISBN: 9781788996013. — URL: https://books.google.ru/books?id=_AhnDwAAQBAJ.

[7] Gupta A. Java EE 7 Essentials: Enterprise Developer Handbook. — O'Reilly Media, 2013. — ISBN: 9781449370602. — URL: https://books.google.ru/books?id=ecJqAAAAQBAJ.

[8] Gutierrez Felipe. Spring Boot, Simplifying Everything // Introducing Spring Framework: A Primer. — Berkeley, CA : Apress, 2014. — P. 263–276. — ISBN: 978-1-4302-6533-7. — URL: https://doi.org/10.1007/978-1-4302-6533-7_19.

[9] JSR 317: Java Persistance 2.0. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://jcp.org/en/jsr/detail?id=317.

[10] Java 8. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.javatpoint.com/java-8-features.

[11] Jones Michael, Campbell Brain, Mortimore Chuck. Json web token (jwt) profile for oauth 2.0 client authentication and authorization grants // May-2015.{Online}. Available: https://tools. ietf. org/html/rfc7523. — 2015.

[12] Keycloak. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.keycloak.org/.

[13] Keycloak: client roles. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.keycloak.org/docs/latest/server_admin/#con-client-roles_server_administration_guide.

[14] Keycloak extensions. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.keycloak.org/docs/latest/server_development/#_extensions.

[15] Keycloak: implementing an authenticator. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.keycloak.org/docs/latest/server_development/#implementing-an-authenticator.

[16] Netflix registration. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.netflix.com/de-en/.

[17] ORM. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://hibernate.org/orm/what-is-an-orm/.

[18] Professional Java Development with the Spring Framework / R. Johnson, J. Höller, A. Arendsen et al. — Wiley, 2007. —

ISBN: 9780471748946. — URL: https://books.google.ru/books?id=oMVIzzKjJCcC.

[19] Pugh E., Gradecki J.D. Professional Hibernate. — Wiley, 2005. — ISBN: 9780764589515. — URL: https://books.google.ru/books?id=Mm7u4p9YjKUC.

[20] Realms. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://docs.oracle.com/cd/E19798-01/821-1841/6nmq2cpjd/index.html.

[21] Reese George. Database Programming with JDBC and JAVA. — " O'Reilly Media, Inc.", 2000.

[22] Salesforce. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.salesforce.com/eu/.

[23] Spring boot mail starter. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://docs.spring.io/spring-framework/docs/current/reference/html/integration.html#mail.

[24] Spring: mock requests. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.baeldung.com/spring-mocking-webclient.

[25] Stricek Art. A reverse proxy is a proxy by any other name // SANS Institute InfoSec Reading Room. — 2002. — Vol. 13.

[26] Twilio: Email API. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://sendgrid.com/solutions/email-api/.

[27] Visualforce. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.tutorialspoint.com/salesforce/salesforce_visualforce_pages.htm.

[28] Wizard, "Волшебный помощник". — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://en.wikipedia.org/wiki/Wizard_(software).

[29] Версии Java. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://en.wikipedia.org/wiki/Java_version_history.

[30] Обратная совместимость Java. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://blogs.oracle.com/java/post/upgrading-major-java-versions.

[31] Регистрация ООО в Тинькофф. — [Электронный ресурс]. — (дата обращения: 06.05.2022). URL: https://www.tinkoff.ru/business/registration-ooo/.