Санкт-Петербургский государственный университет

Кафедра Системного Программирования

Группа 21.Б10-мм

Разработка веб-приложения для обработки банковских транзакций

КСЕНЧИК Никита Владимирович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
доцент кафедры системного программирования, к. т. н., М. А. Серов

Консультант:
менеджер по персоналу ООО «Солантек», В. С. Ветрова

Санкт-Петербург
2023

Оглавление

Введение

В современном информационном обществе, где технологии проникают во все сферы нашей жизни, безналичные платежи занимают особое место. Они стали неотъемлемой частью нашей повседневной рутины, обеспечивая удобство и безопасность при совершении финансовых операций.

Большая востребованность влечет за собой потребность в надежности и высокой скорости работы систем, обеспечивающих возможность проведения платежей. Не каждый банк может позволить себе содержать большую команду программистов, которая будет создавать все необходимое программное обеспечение, постоянно поддерживать и развивать его. Поэтому часто многие банки обращаются за помощью в специализированные компании.

Компания Solanteq – является разработчиком компонентов для построения платежных решений. [7] Solanteq занимается разработкой, внедрением и поддержкой программного обеспечения для обработки транзакций с использованием пластиковых карт, таких как Visa и MasterCard. Их продукт включает в себя функциональность для расчета комиссий и процентов, анализа и предотвращения рисков, а также решения других связанных задач, включая интеграцию с другими банковскими системами, предоставление доступа к нескольким системам через единое окно и проведение функционального и нагрузочного тестирования.

Для новых сотрудников в такой компании начать писать промышленный код очень сложно. Поэтому, в рамках летней стажировки стажерам предлагалось в командах по 2 человека разработать веб-приложение для обработки банковских транзакций, используя технологии из стека компании и интегрируясь с имеющимися модулями Солантека.

1. Постановка задачи

Целью работы является разработка веб-приложения для ручного администрирования клиентов банка и обработки транзакций по соответствующим договорам.

Для достижения этой цели были поставлены следующие задачи:

1. разработать бэкенд части веб приложения:
   (a) разработать модуль, позволяющий удобно взаимодействовать с сущностями базы данных;
   (b) поддержать возможность создания транзакций;
   (c) поддержать обработку подготовленных транзакций;
2. разработать веб интерфейс приложения:
   (a) создать формы для взаимодействия с конфигурационными сущностями;
   (b) создать формы договоров и транзакций, с возможностью создания транзакций на последней;

2. Обзор

2.1. Обзор средств разработки

В проекте использовались из стека компании:

- Сервис GitLab для хранения исходного кода проекта
- YouTrack - инструмент для управления задачами проекта и кооперации разработки с командой
- База данных Postgresql
- Liquibase - это инструмент для управления и автоматизации изменений в базе данных [2]
- Язык Kotlin для реализации backend части приложения
- Для взаимодействия с базой данных из backend части использовался Hibernate
- Spring Framework - это инструментарий для разработки приложений на языке Java/Kotlin
- Mockito - фреймворк для использования мок-объектов в тестировании приложений на языке Java/Kotlin [3]
- Spring Beans — это объекты, которые управляются контейнером Spring и используются для создания и настройки компонентов приложения. Они являются основными строительными блоками приложения, которые можно встраивать (инъектировать) в другие компоненты и использовать для выполнения различных операций [1]

Для UI веб-приложения использовался SOLAR Air. Данный модуль позволяет полностью конструировать интерфейс посредством json файлов (расположение частей интерфейса, действия на форме, источник данных). Также SOLAR Air вместе с модулем SOLAR User Management

предоставляет уже готовый функционал для управления пользователями приложения и логирования ошибок, возникающих в UI части приложения.

3. Общая информация

3.1. Основные понятия используемые в приложении

Центральными понятиями в приложении являются договор и транзакция.

Договор — это абстракция участника финансовых операций. Договор привязан к физическому или юридическому лицу. У каждого договора есть атрибуты, определяющие его. Для договора, представляющего физическое лицо, это, например: ФИО, номер телефона, адрес и т.д. К каждому договору привязаны компоненты баланса, на которых хранятся средства участника финансовых операций. Для каждого компонента баланса существует тип компоненты баланса, подразумевающий тип хранимых на нем средств: собственные средства и заемные средства. Компоненты баланса договора в совокупности определяют баланс договора. Все изменения средств договора происходят путем изменения количества средств на компонентах баланса.

Транзакция — это движение средств между договорами. Выделяются 4 типа транзакций: покупка (purchase), возврат средств (refund), возврат платежа (chargeback), возврат покупки (purchase return). Для каждого типа транзакции существует направление транзакции: дебет или кредит. Дебет подразумевает списание средств с компонента баланса, кредит — начисление.

3.2. Общая характеристика проектируемой системы

Основной идеей веб-приложения для обработки банковских транзакций является обеспечение быстрой и надежной обработки транзакций, а также возможность удобного взаимодействия с ними и с договорами через веб интерфейс.

От веб интерфейса требуется следующие возможности:

- создание, редактирование и удаление договоров, добавление и изменение его атрибутов;

- создание, редактирование и удаление типов атрибутов договоров;
- создание, редактирование и удаление типов компонентов баланса;
- создание, редактирование и архивирование типов транзакций;
- создание транзакций;
- администрирование пользователей приложения.

Необходимо загружать транзакции в базу данных при создании и извлекать их оттуда для дальнейшей обработки.

От бэкенда части приложения требуется:

- возможность запуска приложения с английской или русской локализацией;
- транзакции должны иметь корректный статус на всех этапах обработки;
- обработка транзакции должна происходить независимо от веб интерфейса;

4. Архитектура

4.1. Архитектура приложения

В проекте выделяются 4 основных модуля:

- data - модуль, отвечающий за взаимодействие с базой данных. Все обращения к базе происходят через классы этого модуля;
- forms - модуль, в котором располагаются frontend части приложения и сервисы, к которым frontend обращается напрямую;
- module - модуль, в котором располагается бизнес логика приложения;
- distribution-apps - модуль, содержащий запускаемые приложения:
  - upgrade-tool - приложение, в котором располагаются скрипты и зависимости, для развертывания БД;
  - distribution-server - приложение для обработки банковских транзакций;

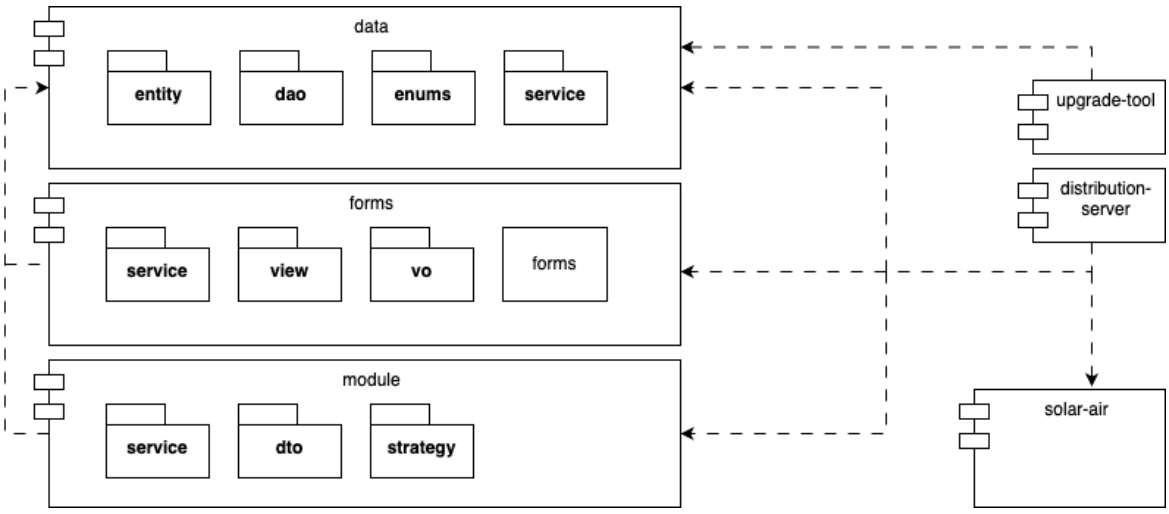

Рис. 1: Структура модулей приложения

4.2. Структура базы данных

Структура базы данных представлена на рисунке 1. База данных состоит из 9 таблиц (за исключением унаследованных SOLAR таблиц для логирования и управления пользователями):

1. **edu_txn_type** - таблица, содержащая типы транзакций;
2. **edu_balance_item_type** - таблица, содержащая типы компонентов баланса;
3. **edu_txn_type__bl_item_type** - таблица, обозначающая отношение "многие ко многим" для таблиц **edu_balance_item_type** и **edu_txn_type**, обозначающая доступные типы компонентов баланса для разных типов транзакций;
4. **edu_attribute_type** - таблица, содержащая информацию о возможных типах аттрибутов договоров;
5. **edu_agreement** - таблица, содержащая информацию о договорах;
6. **edu_agreement_attribute** - таблица, содержащая атрибуты договоров;
7. **edu_balance_item** - таблица, содержащая компоненты баланса;
8. **edu_txn** - таблица, содержащая транзакции;
9. **edu_balance_entry** - таблица, содержащая информацию о движениях средств на компонентах баланса;

В нескольких таблицах присутствуют столбцы audit:

- **audit_date** - дата последнего изменения записи;
- **audit_state** - состояние записи, Active или Removed (для таблиц, записи из которых нужно продолжать хранить после удаления);
- **audit_user_id** - id пользователя, вносившего последнее изменение в конкретную запись;

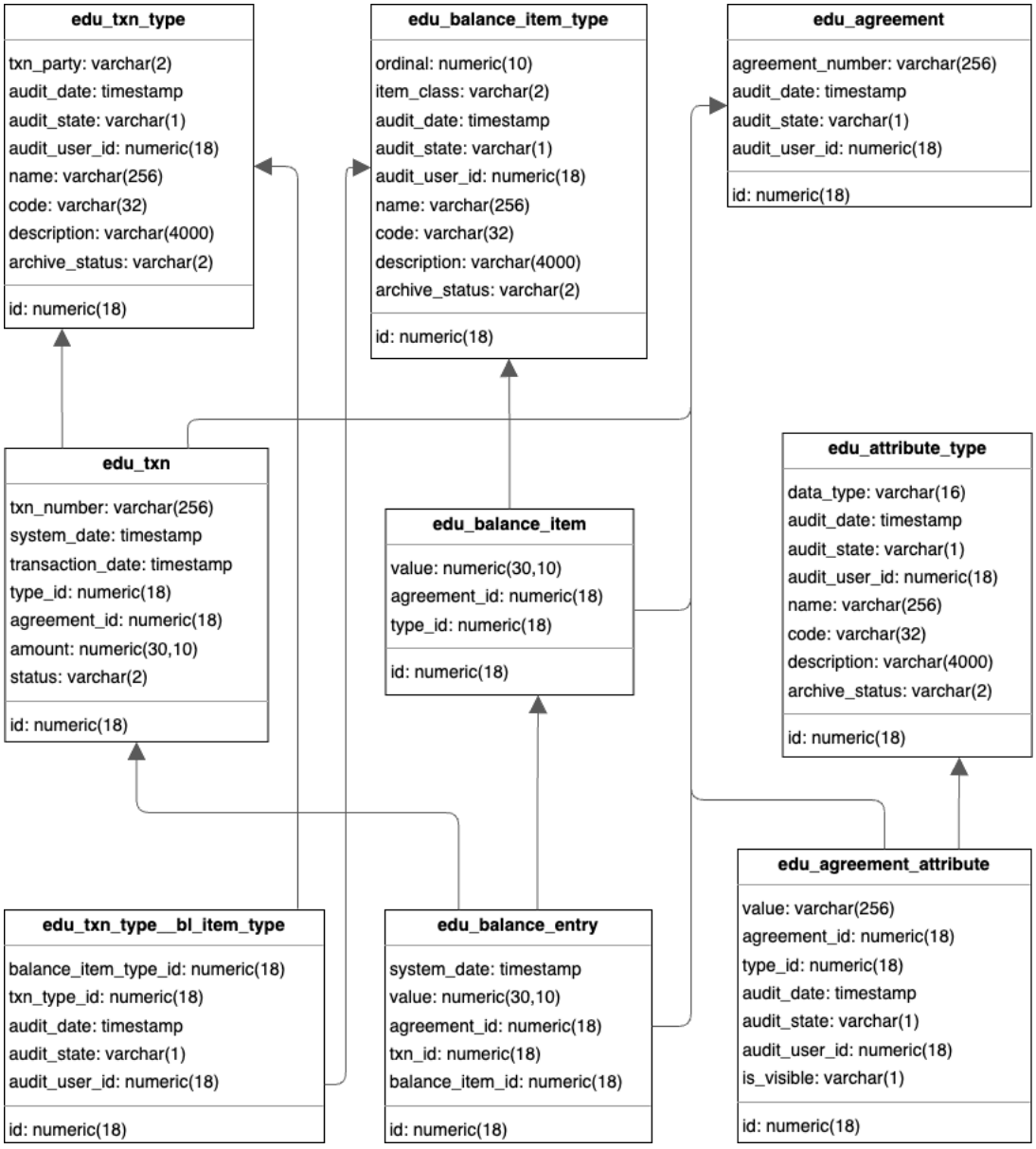

Рис. 2: Структура базы данных

5. Реализация

5.1. Некоторые общие детали реализации

Приложение написано с заделом на дальнейшее расширение и использование RabbitMq. RabbitMq - брокер сообщений который принимает сообщения от отправителей и доставляет их получателям. Приложения зачастую дробятся на обособленные узлы, которые дублируются, для увеличения производительности системы и защиты от сбоев. Для обмена сообщениями между такими узлами и используется RabbitMq. [5]

Основные пакеты присутствующие в проекте:

- entity - классы, используемые как контейнер информации для чтения их и записи в базу данных;
- dao (data acess object) - объекты, используемые как интерфейс для доступа к базе данных;
- service - объекты, используемые в бизнес логике;
- vo (value object) - классы с данными;
- dto (data transfer object) - классы, используемые для передачи данных между слоями приложения;
- view - dto, используемый для передачи данных на frontend;

Все сервисы помечены маркерным интерфейсом SolarEducationService. Данные интерфейс передается в EducationDataAdapter. Это прокси адаптер, позволяющий маршрутизировать запросы, направленные к модулям, помеченным SolarEducationService. В данной реализации всё происходит без RabbitMq, и EducationDataAdapter замещает работу связки адаптера и брокера сообщений.

Веб-интерфейс приложения конфигурируется с помощью json файлов. В таких файлах указываются:

- метаданные о самой форме;

- под-формы данной формы;
- вид данных хранимых в таблицах на форме;
- запрос, по которому запрашиваются данные
- действия, которые можно производить на данной форме (удаление записей, добавление записей, кастомные действия);
- и т.д.

Запрос для получения данных выглядит следующим образом: "<модуль>.<сервис>.<метод>". Приложение знает об имеющихся в нем адаптерах, в них находит информацию, о том, за какие модули отвечает данный адаптер. В нем, среди зарегистрированных сервисов выбирается нужный и вызывается требуемый метод.

5.2. Обработка транзакции

Обработка транзакции начинается с добавления транзакции в начальном состоянии в базу данных. На данный момент в приложении это можно сделать одним способом — через веб интерфейс. На форме создания транзакции необходимо заполнить: тип транзакции, сумму, договор, по которому будет проходить транзакция, транзакция для отмены (для транзакций с типом REFUND), желаемую дату транзакции.

Далее эти данные будут конвертированы в TxnModel, после чего будет вызван метод createTxn. Там будут проверены все имеющиеся данные по создаваемой транзакции, запрошенные необходимые для создания данные из базы, после чего транзакция будет сохранена в базе с нужным статусом: REJECTED для отклонённых транзакций и PREPARED для транзакций прошедших успешно.

На заднем фоне приложения работает BackgroundTxnHandler. Периодически он считывает из базы транзакции со статусом PREPARED, и, если таковые есть, начинает их обработку.

При создании транзакции в онлайне (через ui), чтобы не дожидаться запуска фонового обработчика, созданная транзакция будет вынута из

базы, на неё навешивается lock [4] (для избежания дублирования обработки), и будет начата обработка. Если BackgroundTxnHandler успеет перехватить транзакцию, то поток создавший транзакцию закончит окончания обработки и вернёт уже готовый результат.

Процессирование транзакции проходит по следующим шагам:

1. проверяется, что ожидаемая дата транзакции отсутствует, или меньше текущей
2. проверяется, что транзакция имеет верный статус и верную сумму
3. вешается lock на договор транзакции, во избежания гонок при доступе к компонентам баланса договора
4. выбирается стратегия обработки, в зависимости от типа транзакции
5. происходят движения средств на компонентах баланса, в зависимости от приоритетов компонентов
6. новые данные по компонентам баланса и транзакции сохраняются в базу

Все методы бизнес логики, которые работают с данными из базы данных помечены аннотацией Transactional. Все взаимодействия с базой в таких методах происходят в обособленной транзакции, что позволяет избежать попадания неверных данных в базу при сбоях в работе отдельных методах и всего приложения. [6]

5.3. Веб-интерфейс

Локализация веб-интерфейса доступна на двух языках: русском и английском. Для смены языка требуется перезапуск всего веб-приложения. С учетом возможности работы нескольких приложений параллельно с одной базой данных, предлагается запускать по экземпляру приложения на каждый язык. Первое, что видит пользователь, это форма авторизации. Далее открывается меню, на котором располагаются 3

группы форм: ”Пользователи”, ”Основные формы”, ”Конфигурационные формы”. Формы группы ”Пользователи” добавляются модулем SOLAR Air по умолчанию. Они отвечают за администрирование пользователей, выдачу им прав, изменение паролей, авторизацию при входе.

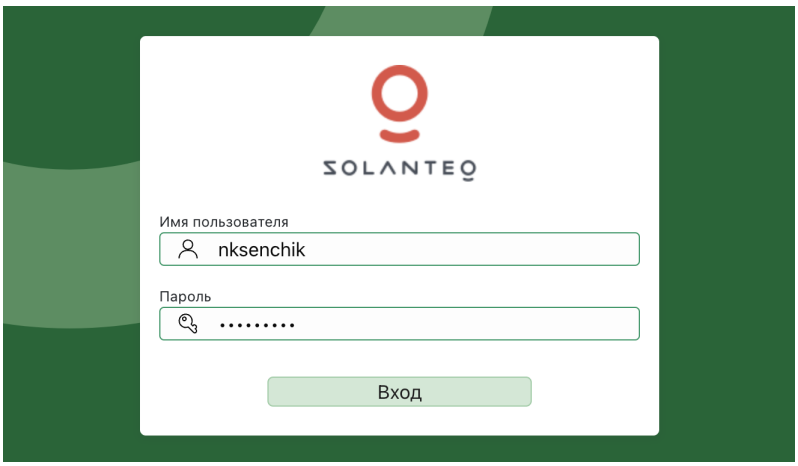

Рис. 3: Форма ”Авторизация”

Рис. 4: Меню

На формах из группы ”Основные формы” и ”Конфигурационные формы” располагаются данные из соответствующих таблиц, доступна сортировка по столбцам, редактирование имеющихся записей, фильтрация записей по разным столбцам, удаление либо архивация записи (конфигурационные сущности только архивируются, а транзакции не архивируются и не удаляются). В группе ”Конфигурационные формы” располагаются ”типы атрибутов”, ”типы компонентов баланса”, ”типы транзакций”.

В группе ”Основных форм” располагаются две формы. Первая - ”Транзакции”. Кроме общих возможностей, на записях этой формы содержатся ссылки на тип транзакции и на детализацию транзакции. На форме детализации, кроме основной информации о транзакции содер-

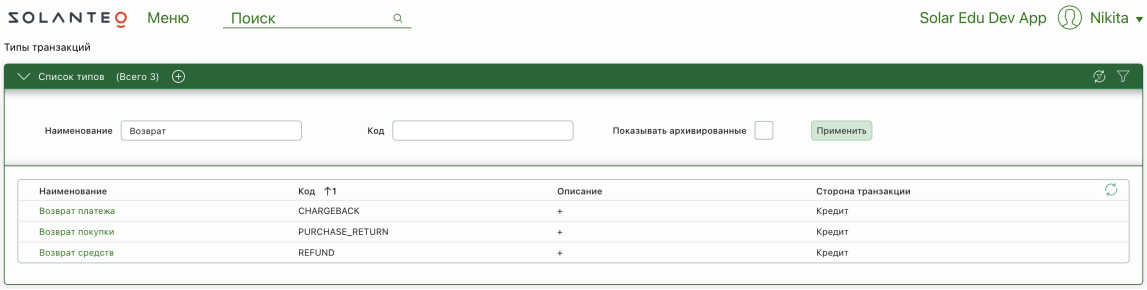

Рис. 5: Форма ”Типы транзакций”

жится дополнительная информация о движениях средств на компонентах баланса, вызванных данной транзакцией.

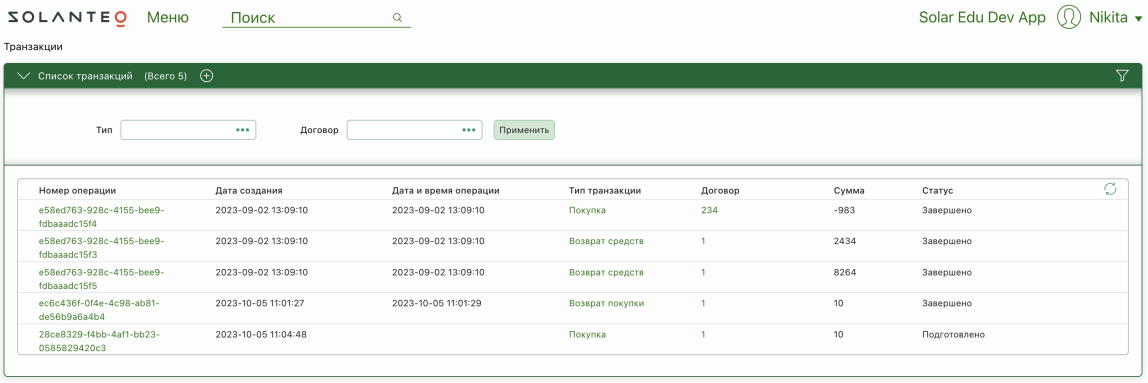

Рис. 6: Форма ”Список транзакций”

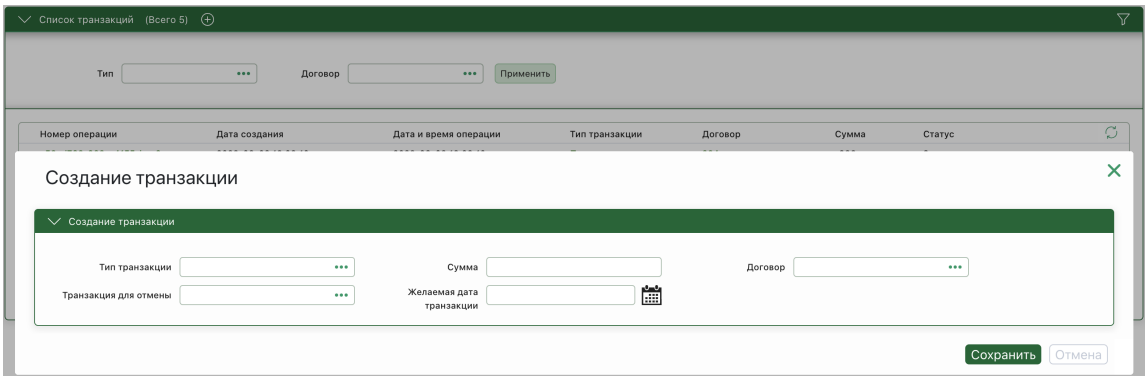

Рис. 7: Действие ”Создание транзакции”

Вторая форма - ”Договоры”. На ней, кроме основного функционала, ссылка на детализацию договора. На ней располагается список атрибутов договора и две вкладки: список транзакций, проходивших по этому договору, и история изменений договора.

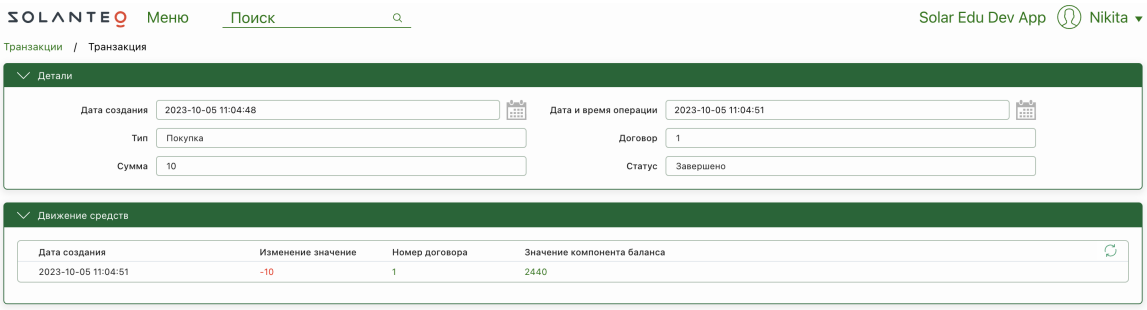

Рис. 8: Форма ”Транзакция”

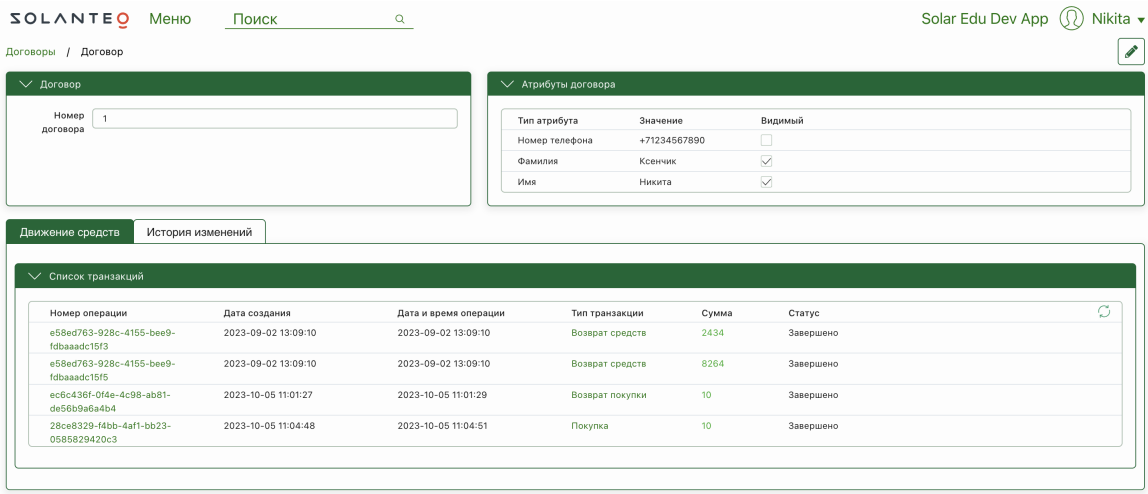

Рис. 9: Форма ”Договор”

5.4. Тесты

Все сервисы, содержащие бизнес логику снабжены Unit тестами.

Заключение

По результатам выполнения практической работы было разработано веб-приложение обработки банковских транзакций.

- разработана backend часть веб-приложения:
    1. разработан модуль, позволяющий удобно взаимодействовать с сущностями базы данных;
    2. поддержана возможность создания транзакций;
    3. поддержана обработка подготовленных транзакций;
- разработан веб-интерфейс приложения:
    1. созданы формы для взаимодействия с конфигурационными сущностями;
    2. созданы формы договоров и транзакций, с возможностью создания транзакций на последней;

Список литературы

[1] Introduction to the Spring IoC Container and Beans. — URL: https://docs.spring.io/spring-framework/reference/core/beans/introduction.html (дата обращения: 8 октября 2023 г.).

[2] Liquibase и Maven. — URL: https://habr.com/ru/articles/436994/ (дата обращения: 8 октября 2023 г.).

[3] Mockito Tutorial. — URL: https://www.digitalocean.com/community/tutorials/mockito-tutorial (дата обращения: 8 октября 2023 г.).

[4] Postgresql, Explicit Locking. — URL: https://www.postgresql.org/docs/current/explicit-locking.html (дата обращения: 8 октября 2023 г.).

[5] RabbitMQ. Часть 1. Introduction. Erlang, AMQP. — URL: https://habr.com/ru/articles/488654/ (дата обращения: 8 октября 2023 г.).

[6] @Transactional в Spring под капотом. — URL: https://habr.com/ru/articles/532000/ (дата обращения: 8 октября 2023 г.).

[7] ООО "Солантек". — URL: https://solanteq.ru/ (дата обращения: 8 октября 2023 г.).