**СОДЕРЖАНИЕ**

Введение5

1 Анализ и моделирование предметной области программного средства 6

1.1 Описание предметной области 6

1.2 Разработка функциональной модели предметной области 8

1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований. 9

1.4 Разработка информационной модели предметной области 11

1.5 Модели представления программного средства и их описание 15

2 Проектирование и конструирование программного средства17

2.1 Постановка задачи17

2.2 Архитектурные решения18

2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства19

2.4 Проектирование пользовательского интерфейса21

2.5 Обоснование выбора компонентов и технологий для реализации программного средства22

2.5 Обоснование выбора компонентов и технологий для реализации программного средства26

3 Тестирование и проверка работоспособности программного средства29

4 Руководство по установке и использованию программного средства31

Заключение33

Список использованных источников34

Приложение А (обязательное)35

Приложение Б (обязательное)36

Приложение В (обязательное)45

**ВВЕДЕНИЕ**

В данной пояснительной записке представлено описание курсового проекта по дисциплине «Программирование сетевых приложений на языке Java».

Тема курсового проекта – это разработка системы Интернет-банкинга и программная поддержка дистанционного обслуживания клиентов. Для этого нужно использовать два клиент-серверных приложения, взаимодействие между которыми будет осуществлено по протоколу TCP\IP. Благодаря этому мы сможем решить задачи различных типов по работе с базами данных. Со стороны клиента будут отправляться запросы на серверную часть приложения. На серверной части, в свою очередь, будет находиться бизнес-логика.

Цель проекта – построение программы на основе клиент-серверной архитектуры с использованием таких технологий как JFreeChart, Socket (TCP), JDBC, Swing, GSON и некоторых других.

Можно разбить имеющуюся цель на несколько задач, выполнение которых необходимо для выполнения поставленной цели.

Таким образом, можно составить задачи на данную курсовую работу:

1 Создать проект с использованием языка программирования Java. Для которого необходимо в данном случае использовать среду разработки IntelliJ IDEA ULTIMATE для того, чтобы можно было использовать более широкий перечень возможностей.

2 Нужно использовать протокол TCP\IP. С его помощью мы реализуем приложение с клиент-серверной архитектурой.

3 Необходимо использовать JDBC драйвера для того, чтобы реализовать серверную часть данного приложения. Для сервера базы данных используем также драйвер Connector\J.

4 Сервером базы данных в данном случае должен выступить MySQL сервер последней версии.

5 Отображение данных в виде таблиц класса JTable с помощью библиотеки Swing необходимо реализовать на клиентской части приложения. С помощью этой библиотеки мы также реализуем графический интерфейс.

6 JFreeChart – библиотека языка Java. Ее необходимо использовать для создания графиков.

7 Необходимо реализовать несколько дополнительных задач, которые помогут более удобно использовать разработанную систему.

8 Составить документацию по разработанной системе с подробным описанием работы программного средства.

**1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА**

* 1. **Описание предметной области**

В данном подразделе пояснительной записки будет описана предметная область текущего курсового проекта.

К настоящему времени сложилась система, когда клиенты банка могут управлять своими счетами, а также осуществлять ряд операций дистанционно. Такая система в целом получила название онлайн-банкинг. Удаленное управление банковскими счетами может осуществляться различными способами:

– посредством телефона – телебанкинг;

– персонального компьютера – РС-банкинг;

– портативных устройств – мобильный банкинг;

– Интернета – Интернет-банкинг.

В настоящее время термин «Интернет-банкинг» полностью завоевал просторы сети Интернет. По своей сути, различие между Интернет-банкингом и традиционной схемой сотрудничества с банком не так велико, как может показаться. И банковские услуги, и счета остались «на своих местах», изменился лишь канал связи между банковскими сотрудниками и их клиентами, точнее, добавился новый: функции, ранее выполняемые исключительно офисом банка, присущи теперь и его веб-сайту. Услуги интернет-банкинга включают в себя следующие возможности:

– осуществлять все коммунальные платежи (электроэнергия, газ, телефон, квартплата, теплоснабжение);

– оплачивать счета за связь (домашний телефон, сотовая связь, интернет) и другие услуги;

– производить денежные переводы, в том числе в иностранной валюте на любой счет в любом банке;

– переводить средства в оплату счетов за товары, в том числе купленные через интернет-магазины;

– покупать и продавать иностранную валюту;

– другие дополнительные услуги: возможность подписки на журналы и газеты, брокерское обслуживание (покупка (продажа) ценных бумаг, создание инвестиционного портфеля, возможность участия в паевых фондах банка.

Главной причиной медленного распространения Интернет-банкинга является относительная небезопасность расчетов и сохранности средств на счетах клиентов. Возможность несанкционированного доступа к чужой информации остается основной проблемой в интернете.

Но в ответ на сложившиеся трудности в интернет-банкинге применяют современные технологии программно-аппаратной защиты, которые постоянно совершенствуются.

Сегодня система Интернет-банкинга применяется в основном для решения следующих основных задач:

– ускорение документооборота, более оперативное, чем в классическом виде, управление денежными средствами;

– экономия времени (нет необходимости ехать в филиал банка и тратить время на дорогу и на простаивание в очередях);

– возможность управления деньгами на расстоянии, то есть когда применение классической технологии невозможно или затруднено по тем или иным причинам.

Конечно, система Интернет-банкинга, несовершенна. Так, к примеру, применение систем электронных платежей сталкивается со многими проблемами, а именно:

– недостаточная степень защиты проведения платежей в Интернете;

– нет единого стандарта на документооборот, правила обработки документов также не стандартизированы;

– технологические схемы прохождения и обработки документации в банках различны;

– большая разница в подготовленности клиентов к работе с вычислительной техникой и системами электронных платежей

Существуют 3 основных решения по реализации транзакции между клиентом и базой данных банка с применением Интернет-технологии:

1 «Голый» WEB. Эта схема попадает под определение «тонкого» клиента. Интерфейс реализован на базе HTML, в качестве протокола HTTP поверх SSI. Клиент использует обычный WEB-браузер. В банке установлен WEB-сервер для исполнения WEB-приложения, который с одной стороны динамически формирует HTML-страницы для клиента, а с другой обращается с сервером базы данных.

2 WEB + программное обеспечение. В данном решении клиенту предлагаются специальные программы или плагин-модули для конкретной версии WEB-браузера. Сложности: проведение установки и настройки специализированного программного обеспечения у клиента и необходимость периодического обновления этого программного обеспечения. Как следствие, необходимость создания в банке группы сотрудников для технической поддержки клиентов и дополнительные издержки банка. Так как программное обеспечение устанавливается у клиента – это система с «толстым» клиентом, а обработка на нашей машине.

3 Применение Java-апплета. Функции клиентской программы выполняет Java-апплет, загружаемый в WEB-браузер клиента. В Java-апплете реализован весь интерфейс пользователя: экранные и печатные формы документов, проверки правильности заполнения, протокол защищенного взаимодействия с сервером базы данных, шифрование данных, генерация криптоключей, механизм электронно-цифровой подписи клиента под финансовыми документами и обмен финансовыми документами с автоматизированными банковскими системами.

Независимо от используемой схемы, определяют состав Интернет-банкинга:

1 Клиентская часть системы. Это интернет-сервер, устанавливаемый в банке, который посещают клиенты банка для выполнения операций в системе и где реализован прокол защищенного взаимодействия и шифрования данных.

2 Сервер базы данных. Хранит все документы и открытые ключи электронно-цифровой подписи клиентов, всю информацию о клиентах и справочники. В данной части системы происходит первичная регистрация клиентов, определение счетов и полномочий. Хранятся справочники, используемые клиентом в работе и полная информация о клиенте.

3 Шлюз к антиблокировочной системе. Обеспечивает обмен данными между системами. Обычно поддерживается работа антиблокировочной системы в пакетном режиме, режиме реального времени, может быть комбинировано. Самым распространенным является обмен текстовыми файлами заранее определенного формата.

Для белорусского рынка традиционным решением является: на стороне клиента устанавливается специальное программное обеспечение, дополнительно плагин-модули, а иногда и аппаратные средства. Для работы системы с клиентом используются WEB-браузеры.

Таким образом, реальный эффект от применения системы Интернет-банкинга складывается из многих показателей. Важны не только качественные и количественные показатели системы как таковой, а способ ее применения, то есть возможности, которые реализует и поддерживает банк.

* 1. **Разработка функциональной модели предметной области**

IDEF0 – нотация описания бизнес-процессов. Основана на методологии SADT. SADT – графические обозначения и подход к описанию систем. Разработка SADT началась в 1969 году и была опробована на практике в компаниях различных отраслей (аэрокосмическая отрасль, телефония и так далее). Публично появилась на рынке в 1975 г. и получила очень широкое распространение в мире.

IDEF0 является результатом программы компьютеризации промышленности, которая была предложена ВВС США. Автоматизация деятельности предприятий потребовала соответствующих методик и инструментов. Перед тем, как разрабатывать программное обеспечение, необходимо было четко и понятно описать бизнес-процессы (нельзя автоматизировать хаос). Инструменты, разработанные для задач программирования, так же могут быть полезны и для задач менеджмента.

Нотация может быть использована для моделирования широкого круга автоматизированных и неавтоматизированных систем.

Идея IDEF0 лежит в том, что бизнес-процесс отображается в виде прямоугольника, из которого входят и выходят стрелки.

Для IDEF0 имеет значение сторона процесса и связанная с ней стрелка:

– слева входящая стрелка – вход бизнес-процесса – информация (документ) или товарно-материальная ценность, которая будет преобразована в ходе выполнения процесса;

– справа исходящая стрелка – выход бизнес-процесса – преобразованная информация (документ) или товарно-материальная ценность;

– сверху входящая стрелка – управление бизнес-процесса – информация или документ, который определяет, как должен выполняться бизнес-процесс, как должно происходить преобразование входа в выход;

– снизу входящая стрелка – механизм бизнес-процесса – то, что преобразовывает вход в выход: сотрудники или техника. Считается, что за один цикл процесса не происходит изменения механизма.

Выход одного бизнес-процесса является входом / управлением / механизмом другого бизнес-процесса. На диаграмме процессы принято располагать по диагонали с верхнего левого угла в нижний правый угол. Количество процессов не более 6-8.

Преимущество IDEF0 показывает взаимодействие процессов в общем виде, без лишних подробностей.

Недостаток IDEF0 – нельзя увидеть алгоритма выполнения бизнес-процессов, требует определенной подготовки для разработки и чтения нотации.

Основными потребителями нотации IDEF0 являются руководители, которым необходимо видеть и понимать взаимосвязь процессов, не вникая в мелочи.

Верхним уровнем в данном случае является процесс организации системы Интернет-банкинга. Данное окно представлено на рисунке 1.1.

* 1. **Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Перед тем как начать разрабатывать программное средство необходимо проанализировать требования к программному средству и поставить задачи. Цель процесса анализа требований к программному средству заключается в установлении и документировании требований к программному обеспечению.

Разработка программного средства будет вестись на операционной системе Windows 10. При разработке программного средства будут использованы следующие технологии: Java и БД MySQL.

Каждая из выбранных технологий отвечает за разные аспекты работы программы.

Основным инструментом для создания проекта стала среда – IntelliJ IDEA Ultimate.

**IntelliJ IDEA** – это ведущая среда быстрой разработки на языке Java. IntelliJ IDEA представляет собой высокотехнологичный комплекс тесно интегрированных инструментов программирования, включающий интеллектуальный редактор исходных текстов с развитыми средствами автоматизации, мощные инструменты рефакторинга кода, встроенную поддержку технологий J2EE, механизмы интеграции со средой тестирования Ant/JUnit и системами управления версиями, уникальный инструмент оптимизации и проверки кода Code Inspection, а также инновационный визуальный конструктор графических интерфейсов.

Уникальные возможности JetBrains IntelliJ IDEA избавляют программиста от груза рутинной работы, помогают своевременно устранить ошибки и повысить качество кода, поднимая продуктивность разработчика на новую высоту.

Первая версия появилась в январе 2001 года и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ.

Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

В редакции Ultimate Edition реализована поддержка Java EE, UML-диаграмм, подсчёт покрытия кода, а также поддержка других систем управления версиями, языков и фреймворков.

Поддерживаемые языки:

– Java;

– JavaScript;

– CoffeeScript;

– HTML/XHTML/HAML;

– CSS/SASS/LESS;

– многие другие.

Notepad++ – свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса большого количества языков программирования и разметки. Проект находился на SourceForge.net, откуда он был скачан (по состоянию на 23 июля 2015 года) более 29 миллионов раз.

Maven – это инструмент для сборки Java проекта: компиляции, создания jar, создания дистрибутива программы, генерации документации. Простые проекты можно собрать в командной строке.

Если собирать большие проекты с командной строки, то команда для сборки будет очень длинной, поэтому её иногда записывают в bat/sh скрипт. Но такие скрипты зависят от платформы. Для того чтобы избавиться от этой зависимости и упростить написание скрипта используют инструменты для сборки проекта.

Workbench – это средство для более удобного создания базы данных посредствам, например, использования конструктора.

Так как мы используем данные технологии и инструменты мы имеем возможность существенно сократить время на создание данного проекта.

Все перечисленные выше технологии, а также соответствующий инструментарий идеально подходят для разработки приложений.

Любой цикл разработки программного средства начинается с анализа требований. Цель этой стадии – определение детальных требований к программному средству.

Спецификация требований к программному обеспечению – это описание поведения системы, которую необходимо разработать.

Спецификация функциональных требований включает пользовательские сценарии, которые носят название варианты использования.

Каждый вариант использования – это некоторое действие, которое доступно некоторому человеку, или актёру. То есть, это некоторый логически завершённый вариант работы системы.

Актёр – это некоторая обобщённая роль человека в системе, характеризующаяся конкретными привилегиями и возможностями в данной системе.

Актёры и варианты использования – основы диаграммы вариантов использования.

Диаграмма вариантов использования (Use Case) в UML – это диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – это возможность моделируемой системы, благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.

Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Для данного проекта диаграмма вариантов использования представлена на рисунке 1.6.

**1.4 Разработка информационной модели предметной области**

Сначала необходимо создать информационную модель, чтобы максимально подробно отобразить суть в базе данных SQL. Набор различных таблиц в базе данных, их связи и атрибуты являются представлением информационной модели.

Предметная область – часть реального мира, рассматриваемая в пределах данного контекста. Под контекстом здесь может пониматься, например, область исследования или область, которая является объектом некоторой деятельности.

Предметная область информационной системы рассматривается как совокупность реальных процессов и объектов (сущностей), представляющих интерес для её пользователей.

У всех объектов имеется определённый набор атрибутов, среди них выделяют существенные и малозначительные. Для того, чтобы упростить процедуру формализации предметной область часто прибегают к процессу дробления всего множества объектов предметной области на мелкие подгруппы объектов, однородных по своей структуре и поведению. Они называются типами объектов.

Экземпляры такого рода объектов имеют одинаковый набор атрибутов, но отличаются значением хотя бы одного атрибута.

Каждому объекту присваивается определённый идентификатор – ключевой атрибут или набор атрибутов. Такие идентификаторы называются первичным ключом. Его значение всегда будет является уникальным и необходимым.

Информационные потребности всей совокупности пользователей будущей системы определяются этапе анализа предметной области информационной системы. Которые, в свою очередь, предопределяют информацию, которая будет храниться в базах данных.

Параметрами предметной области может быть всё что угодно. Например: программное обеспечение, персональный компьютер, группа каких-нибудь лиц (одноклассники, одногруппники, товарищи по интересам), конкретное лицо, различного рода изобретения, их создатели, университетские преподаватели, студенты, используемая ими программа или набор программ, имеющиеся у них девайсы(телефоны, планшеты), стол, стул, бутылка, кружка, пенал, зарядное устройство, мусорное ведро, тапочки, полотенце, шампунь, очки, штаны, ноутбук, бумажный лист, шариковая ручка, дверная ручка, эспандер, миска, кровать, подушка, одеяло, пододеяльник, простынь, наволочка, покрывало, утюг, шторы, компьютерная мышка, динамики, удлинитель, розетка, тумбочка, крышка от бутылки, коробка от пиццы, лампочка, плафон, пожарный извещатель, люстра, лицо, рука, нога, голова, палец, здание, кирпич, стена, фундамент, пол, потолок, лестница, поручень, окно, оконная рама, дверь, шуруп, гайка, болт. У всех этих примеров есть определённые атрибуты (набор свойств).

Информационный объект – это описание некоторой сущности (реального объекта, явления, процесса, события) в виде совокупности логически связанных реквизитов (информационных элементов). Такими сущностями для информационных объектов могут служить: цех, склад, материал, вуз, студент, сдача экзаменов и т.д.

Между объектами предметной области могут существовать связи, имеющие различный содержательный смысл. Существует два типа таких связей: обязательные или факультативные.

Далее будет рассмотрено приведение к третьей нормальной форме модели.

Третья нормальная форма – одна из возможных нормальных форм отношения в реляционной базе данных. 3NF была изначально сформулирована в 1971 году.

Реляционная база данных содержит как структурную, так и семантическую информацию. Структура базы данных определяется числом и видом включенных в нее отношений, и связями типа «один ко многим», существующими между кортежами этих отношений. Семантическая часть описывает множество функциональных зависимостей, существующих между атрибутами этих отношений.

Можно сказать, что функциональные зависимости представляют собой связи типа «один ко многим», существующие внутри отношения.

Некоторые функциональные зависимости могут быть нежелательны.

Корректной считается такая схема базы данных, в которой отсутствуют избыточные функциональные зависимости. В противном случае приходится прибегать к процедуре декомпозиции (разложения) имеющегося множества отношений. При этом порождаемое множество содержит большее число отношений, которые являются проекциями отношений исходного множества. Обратимый пошаговый процесс замены данной совокупности отношений другой схемой с устранением избыточных функциональных зависимостей называется нормализацией.

Условие зависимости от «полного ключа» неключевых атрибутов обеспечивает то, что отношение находится во второй нормальной форме; а условие зависимости их от «ничего, кроме ключа» – то, что они находятся в третьей нормальной форме.

То есть если все атрибуты являются простыми, а все домены содержать исключительно скалярные значения, то можно сказать, что отношение находится в первой нормальной форме. Должны быть исключены все повторения строк в конкретной таблице.

Если в определённой таблице в какой-то из имеющихся ячеек храниться более одного значения или несколько строк с одним и тем же, значением-то данная база данных в первой нормальной форме не находиться.

В реляционной модели отношение всегда находится в первой нормальной форме. А MySQL является реляционной базой данных. Таким образом приходим к выводу о том, что в нашей модели все данные находятся в первой нормальной форме.

В случае если отношение находится в первой нормальной форме и все не ключевые атрибуты зависят от первичного ключа, то данное отношение находится во второй нормальной форме.

Отношение находится в третьей нормальной форме, когда находится во второй нормальной форме и каждый не ключевой атрибут зависит от первичного ключа.

Текущая информационная модель представлена на рисунке 1.7.

Данная модель состоит из шести таблиц.

Данные таблицы описывают данное программное средство и связанные с ним параметры.

Иначе говоря, какому-то банку необходима система Интернет-банкинга и программная поддержка дистанционного обслуживания клиентов. Есть определённый ряд параметров. Основываясь на этих параметрах, данная система будет всё обрабатывать и вносить в базу данных.

Таблица «пользователи» предназначена для хранения информации о пользователях, которые пользуются данной системой в данный момент или пользовались ею раньше. В данной таблице хранятся следующие данные:

– ид – первичный ключ;

– имя;

– фамилия;

– логин;

– пароль;

– номер телефона

– заблокированная информация.

Таблица «транзакции» предназначена для хранения информации о транзакциях денежных средств. В этой таблице хранятся следующие данные:

– ид – первичный ключ;

– комментарий;

– сумма;

– дата.

Таблица «валюта» предназначена для хранения информации о том, в какой валюте были произведены операции. В этой таблице хранятся следующие данные:

– ид – первичный ключ;

– код валюты;

– название валюты.

Таблица «аккаунт компании» предназначена для хранения информации о данных аккаунта компании. В этой таблице хранятся следующие данные:

– ид – первичный ключ;

– сумма;

– логин аккаунта компании;

– пароль аккаунта компании.

Таблица «полномочия» предназначена для хранения информации о полномочиях пользователя. В этой таблице хранятся следующие данные:

– ид – первичный ключ;

– полномочия.

Таблица «тип» предназначена для хранения информации о типе транзакции. В этой таблице хранятся следующие данные:

– ид – первичный ключ;

– категория;

– название.

Так как все параметры данных таблиц зависят только от id, который является первичным ключом в каждой таблице, то можно сделать вывод, что таблица находится в третьей нормальной форме.

Исходя из вышесказанного можно сделать вывод о том, что данная таблица находятся в третьей нормальной форме, образуя при этом целостную и самодостаточную систему, на которой и строится данный курсовой проект.

**1.5 Модели представления программного средства и их описание**

Модели представления системы представляют собой некоторое формальное описание системы посредствам некоторых блок-схем, диаграмм или прочих способов иллюстрирования.

Соответственно, данный подраздел и посвящён описанию этих моделей представления информации.

Первая диаграмма – это диаграмма состояний.

Диаграмма состояний – это диаграмма, которая определяет множество систем: от компьютерных программ до бизнес-процессов.

Условные обозначения:

1 Круг – обозначает начальное состояние.

2 Окружность с маленьким кругом внутри – обозначает конечное состояние (если есть).

3 Скруглённый прямоугольник – обозначает состояние. Верхушка прямоугольника содержит название состояния. В середине может быть горизонтальная линия, под которой записываются активности, происходящие в данном состоянии.

4 Стрелка – обозначает переход. Название события (если есть), вызывающего переход, отмечается рядом со стрелкой.

5 Толстая горизонтальная линия с либо множеством входящих линий и одной выходящей, либо одной входящей линией и множеством выходящих – обозначает объединение и разветвление соответственно.

На данной диаграмме представлен один из основных процессов – выполнение операций в системе. Первое состояние – подключение к серверу. После этого события есть выбор, или успех подключения или ошибка подключения. После этого опять же можно либо выйти, либо перейти к выполнению операций с таблицей. Затем возвращаемся к просмотру таблицы и начинаем цикл данного события.

Далее следует рассмотреть диаграмму классов.

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

На данной диаграмме классов представлены классы из папки src/main/java/data\_base.

Диаграмма классов является ключевым элементом в объектно-ориентированном моделировании. На диаграмме классы представлены в рамках, содержащих три компонента:

1 В верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом. Имена классов начинаются с заглавной буквы. Если класс абстрактный — то его имя пишется полужирным курсивом.

2 Посередине располагаются поля (атрибуты) класса. Они выровнены по левому краю и начинаются с маленькой буквы.

3 Нижняя часть содержит методы класса. Они также выровнены по левому краю и пишутся с маленькой буквы.

**2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

**2.1 Постановка задачи**

Основной задачей данного курсового проекта является создание системы Интернет-банкинга и программная поддержка дистанционного обслуживания клиентов.

Что касается подзадач курсового проекта, то выделяют следующие:

– проектирование функциональной модели системы в нотации IDEF0;

– проектирование и разработка информационной модели в нотации IDEF1X;

– описание спецификации программного средства с помощью UML- диаграммы вариантов использования системы;

– описание структуры программного средства посредством диаграммы компонентов, диаграммы развертывания и диаграммы последовательности;

– описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства;

– тестирование программного средства;

– составление руководства пользователя.

Следующей стадией разработки программного средства будет проектирование. На этом этапе для упрощения визуализации процесса проектирования используются так называемые нотации – схематическое выражение характеристик разрабатываемого программного средства.

После того, как программное средство будет спроектировано, произойдет переход к следующей стадии жизненного цикла – непосредственно разработке. На этом этапе будет разрабатываться программное средство в соответствии с ранее определенными требованиями.

После того, как этап разработки программного средства будет завершен, необходимо будет протестировать полученное программное средство. На этом этапе необходимо будет найти дефекты в программном средстве и сравнить описанное в требованиях поведение системы с реальным.

При обнаружении дефекта, необходимо будет его исправить, после чего повторить тестирование и на этот раз убедиться, что проблема исправлена и само исправление не стало причиной появления новых дефектов в системе.

После того, как тестирование программного средства пройдет успешно, необходимо будет пройти этап документации. Он будет включать в себя составление руководства пользователя, а также презентацию программного средства.

Выполнение всех описанных выше стадий жизненного цикла программного средства является обязательным при проектировании, разработке, тестировании и запуске программного средства.

**2.2 Архитектурные решения**

Многие программисты решают идентичные задачи и находят похожие решения. Для того, чтобы не создавать проблем при проектировании, можно воспользоваться уже готовыми решениями – фреймворками и шаблонами проектирования. Spring наиболее часто используемая Java EE Framework и Hibernate наиболее популярная реализация ORM. Поэтому комбинация Spring Hibernate часто используется в корпоративных приложениях.

Под технологиями в данном случае имеется в виду комбинация паттернов проектирования и фреймворков. Целью работы является демонстрация применения технологий для тех, кто впервые столкнулся с проблемами доступа к данным из java-приложений. В книге описаны паттерны доступа к данным, на которых построены фреймворки, применяемые для создания корпоративных приложений.

Паттерн – это архитектурная конструкция, помогающая описать и решить некую задачу проектирования. Ко времени представления паттернов разработка программного обеспечения (ПО) стала индустриальной задачей. Многие программисты понимали, что не стоит изобретать велосипед при создании нового ПО. Использование паттернов часто помогает решить эту задачу и бывает полезным, как отдельному разработчику, так и целой команде.

Java – это объектно-ориентированный язык программирования, созданный в 1991 году. Изначально данный язык использовался в основном для бытовой электроники, но впоследствии стал использоваться для написания приложений и серверного программного обеспечения.

С момента представления в 1995 г. Java-платформы как единого целого мира, Java прошел радикальный эволюционный путь от концепции «апплеты повсюду», которую исповедовали первые идеологи и приверженцы.

Вместо этого мир Java развился до Swing, сконцентрировался вокруг сервлетов, направил движение к J2EE, споткнулся на EJB, нашел обходной путь через Spring и Hibernate, добавил возможности абстрактного программирования и стал более динамичным, а затем и более функциональным, и продолжает развиваться во множестве интересных направлений, в то время как я пишу эту статью.

Это многообразие может несколько озадачить Java-программиста, если он не рос и не развивался профессионально вместе с данным языком все эти годы.

Разработчики языка JavaScript использовали популярность активно развивающегося языка Java и на его основе дали такое название своей технологии.

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) – программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной.

Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят снижение производительности.

Идеи, заложенные в концепцию и различные реализации среды виртуальной машины Java, вдохновили множество энтузиастов на расширение перечня языков, которые могли бы быть использованы для создания программ, исполняемых на виртуальной машине. Эти идеи нашли также выражение в спецификации общеязыковой инфраструктуры CLI, заложенной в основу платформы .NET компанией Microsoft.

Следующие успешные проекты реализованы с привлечением Java технологий: RuneScape, Amazon, eBay, LinkedIn, Yahoo!

Следующие компании в основном фокусируются на Java технологиях: SAP, IBM, Oracle. В частности, СУБД Oracle Database включает JVM как свою составную часть, обеспечивающую возможность непосредственного программирования СУБД на языке Java, включая, например, хранимые процедуры.

**2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства**

В рамках данного подпункта будет рассмотрено построение и реализация алгоритмов, реализующих бизнес-логику. Рассмотрим алгоритмы построения некоторых методов различных классов. Будут приведены их блок-схемы реализации.

Первый метод, алгоритм которого будет рассмотрен здесь – это метод setPortNumber класса BAMain серверной части данного приложения.

Данный метод производит обработку введённого номера порта.

Как уже говорилось ранее, данный метод является private-методом класса BAMain серверного приложения, реализующий бизнес-логику, листинг кода данного метода будет представлен далее:

private void setPortNumber() throws Exception {

String portStr = mainWindow.textFieldList.get(0).getText();

try {

port = Integer.valueOf(portStr);   
} catch (Exception e) {

throw new Exception("ВВЕДИТЕ ЧИСЛО");   
}   
if (port < 1500 || port > 60000) {

throw new Exception("ВВЕДИТЕ ПОРТ В ДИАПАЗОНЕ 1500..60000");

}

}

Представлен данный метод на рисунке 2.1.

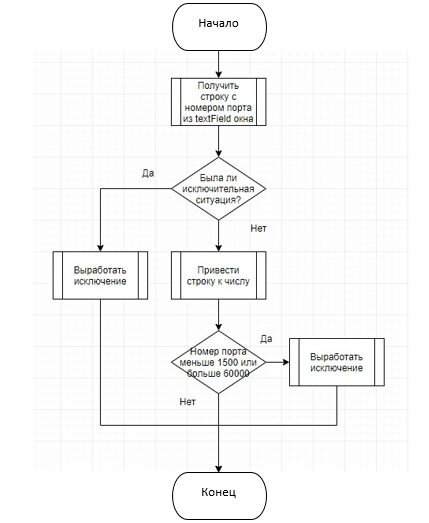


Рисунок 2.1 – Блок-схема алгоритма метода setPortNumber

Видим, что метод не принимает входные параметры. А также данный метод имеет тип возвращаемого значения void.

Сначала необходимо получить строку с номером порта из текстового поля. На блок-схеме это обозначено блоком вызова внешней функции.

Часть кода обёрнута в try-catch конструкцию. Это говорит о том, что может быть выработано исключение. Это отображено на блок-схеме в виде первой ветвящейся конструкции. Если исключение сгенерировано – одно действие, если нет – другое.

Блок try. Сначала мы приводим строку к числу. Данное действие обозначено блоком вызова внешней функции.

Блок catch. В данном блоке лишь одно действие. Генерируется исключение с текстом ошибки, в случае если строка не являлась числом.

После блока try-catch имеется ещё одна конструкция – конструкция if.

В данной конструкции проверяется, входит ли порт в допустимый диапазон. Если нет, то генерируется исключение с текстом ошибки. Если да, то функция завершает свою работу.

На этом описание данного метода можно считать полностью завершенным.

Можно сделать вывод, что данный подраздел позволил описать блок-схему метода, реализующего бизнес-логику, сделать это подробно и с применением блок-схем алгоритма. Это помогло досконально и точно описать всю структуру данного метода.

**2.4 Проектирование пользовательского интерфейса**

Пользовательский интерфейс на Java прошел весьма тернистый путь становления и развития. Долгое время его обвиняли в медленной работе, жадности к ресурсам системы, ограниченной функциональности. Появление .NET с более быстрыми графическими компонентами еще больше пошатнуло позиции Java.

AWT была первой попыткой Sun создать графический интерфейс для Java. Они пошли легким путем и просто сделали прослойку на Java, которая вызывает методы из библиотек, написанных на С. Библиотечные методы создают и используют графические компоненты операционной среды. С одной стороны, это хорошо, так как программа на Java похожа на остальные программы в рамках данной ОС. Но с другой стороны, нет никакой гарантии, что различия в размерах компонентов и шрифтах не испортят внешний вид программы при запуске ее на другой платформе. Кроме того, чтобы обеспечить мультиплатформенность, пришлось унифицировать интерфейсы вызовов компонентов, из-за чего их функциональность получилась немного урезанной. Да и набор компонентов получился довольно небольшой. К примеру, в AWT нет таблиц, а в кнопках не поддерживается отображение иконок.

Использованные ресурсы AWT старается освобождать автоматически. Это немного усложняет архитектуру и влияет на производительность. Освоить AWT довольно просто, но написать что-то сложное будет несколько затруднительно. Сейчас ее используют разве что для апплетов.

В настоящее время AWT используется крайне редко – в основном в старых проектах и апплетах.

Вслед за AWT Sun разработала набор графических компонентов под названием Swing. Компоненты Swing полностью написаны на Java. Для отрисовки используется 2D, что принесло с собой сразу несколько преимуществ. Набор стандартных компонентов значительно превосходит AWT по разнообразию и функциональности. Стало легко создавать новые компоненты, наследуясь от существующих и рисуя все, что душе угодно. Стала возможной поддержка различных стилей и скинов. Вместе с тем скорость работы первых версий Swing оставляла желать лучшего. Некорректно написанная программа и вовсе могла повесить винду намертво.

Тем не менее благодаря простоте использования, богатой документации и гибкости компонентов Swing стал, пожалуй, самым популярным графическим фреймворком в Java. На его базе появилось много расширений, таких как SwingX, JGoodies, которые значительно упрощают создание сложных пользовательских интерфейсов. Практически все популярные среды программирования Java включают графические редакторы для Swing-форм. Поэтому разобраться и начать использовать Swing не составит особого труда.

**2.5 Обоснование выбора компонентов и технологий для реализации программного средства**

В данной работе будет использован широкий перечень технологии. Разберём их по порядку.

Первой технологией является язык программирования – Java.

В данной курсовой работе будет использоваться редакция языка Java – Java SE.

Java SE (Java Platform Standard Edition) – это стандартная редакция Java, которая используется для разработки простых Java приложений. Например, можно создавать консольные приложения, апплеты, приложения с графическим интерфейсом. До 6 версии Java эта редакция имела название J2SE (Java 2 Platform Standard Edition).

Для создания графиков используется библиотека JFreeChart языка Java.

Сегодня библиотека JFreeChart широко используется в Java приложениях для создания широкого спектра графиков. Используя JFreeChart можно создавать все основные типы 2D и 3D графики: круговые диаграммы, гистограммы, линейные и временные диаграммы. Библиотека позволяет создавать изображения нескольких форматов типа PNG, JPEG, SVG и т. д.

Для управления БД используется MySQL.

MySQL – свободная система управления базами данных (СУБД). MySQL является собственностью компании Oracle Corporation, получившей её вместе с поглощённой Sun Microsystems, осуществляющей разработку и поддержку приложения. Распространяется под GNU General Public License или под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей, именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

26 февраля 2008 года Sun Microsystems приобрела MySQL AB за 1 миллиард долларов. 27 января 2010 года Oracle Corporation приобрела Sun Microsystems и включила MySQL в свою линейку СУБД.

Сообществом разработчиков MySQL созданы различные ответвления кода, такие как Drizzle, OurDelta, Percona Server, и MariaDB. Все эти ответвления уже существовали на момент поглощения компаний Sun и MySQL AB корпорацией Oracle.

А теперь перечислим преимущества MySQL.

1 Быстродействие. Благодаря внутреннему механизму многопоточности быстродействие MySQL весьма высоко.

2 Безопасность. Довольно высокий уровень безопасности обеспечивается благодаря базе данных mysql, создающейся при установке пакета и содержащей пять таблиц. При помощи этих таблиц можно описать, какой пользователь из какого домена с какой таблицей может работать и какие команды он может применять. Пароли, хранящиеся в базе данных, можно зашифровать при помощи встроенной в MySQL функции password().

3 Лицензия. Раньше лицензирование MySQL было немного запутанным; сейчас эта программа для некоммерческих целей распространяется бесплатно.

4 Открытость кода. Благодаря этому вы сможете сами добавлять в пакет нужные функции, расширяя его функциональность так, как вам требуется. Кстати, за отдельную плату для вас это могут сделать и сами авторы MySQL.

5 Надежность. Создатели MySQL потрудились на славу: насколько мне известно, этот пакет довольно стабилен и его трудно вывести из строя. Я не отслеживаю специально сводки результатов хакерских атак на MySQL, но мне ни разу не попадалось на глаза (в отличие от тех же Web-серверов) сообщение о том, что MySQL был поврежден в результате чьего-то злого умысла.

6 Ресурсы. Это может зависеть от разных факторов, но в любом случае суперкомпьютер вам не потребуется.

7 Сообщество. Как следствие открытости кода, бесплатности программы, стабильной и надежной ее работы образовалось сообщество людей, которые не просто лояльны к MySQL, но и всячески участвуют как в развитии самого пакета, так и в обучении менее опытных людей работе с ним. Существует огромное количество листов рассылки и конференций, где можно получить бесплатную помощь в любое время суток.

8 Переносимость. В настоящее время существуют версии программы для большинства распространенных компьютерных платформ. Это говорит о том, что вам не навязывают определенную операционную систему. Вы сами можете выбрать, с чем работать, например, с Linux или Windows, но даже в случае замены ОС вы не потеряете свои данные и вам даже не понадобятся дополнительные инструменты для их переноса.

Не знаю, к недостаткам или преимуществам отнести тот факт, что у MySQL нет графического интерфейса пользователя (GUI). Мне, например, удобнее написать SQL-запрос вручную (кстати, результаты его выполнения можно перенаправить в файл), чем пользоваться мастером запросов, как в Microsoft SQL Server.

Существует несколько клиентских программ для MySQL, имеющих GUI, но они далеки от совершенства и по большей части только тормозят работу. Если вы предпочитаете GUI, то рекомендую скачать и попробовать эти программы, чтобы подтолкнуть их создателей к дальнейшему усовершенствованию своих изделий. Вот далеко не полный список программ с GUI:

1 Winmysqladmin – входит в Windows-дистрибутив MySQL, имеет стандартный графический интерфейс и позволяет администрировать MySQL.

2 Входит в Windows-дистрибутив MySQL, несет в себе клиентские функции (но ничего серьезного с ее помощью автору сделать так и не удалось).

3 MySQL Administrator for Windows – более "продвинутая" утилита сторонних разработчиков. Позволяет зарегистрировать и подключаться одновременно к нескольким MySQL-серверам, создавать, удалять и изменять структуру баз данных и таблиц, создавать в таблицах ключи, писать SQL-запросы и сохранять их в файле.

4 XMySQL – клиент MySQL для X Window-подобных систем. Предоставляет полный доступ к таблицам, допускает групповые вставки и удаления, имеет конструктор запросов и функции администрирования пакета.

Для проектирования программного средства используется язык графического описания UML.

UML – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, н основании UML-моделей возможна генерация кода.

Следующим этапом данного пункта будет рассмотрение структуры программы посредствам следующих UML-диаграмм:

– диаграмма развертывания и компонентов;

– диаграмма последовательности.

Диаграмма компонентов – это элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами.

В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и другое.

Диаграмма развёртывания в UML моделирует физическое развертывание артефактов на узлах.

Данная диаграмма отражает основные физические машины, на которых работает система, а также приложения в них для текущей системы (основные компоненты).

Диаграмма развертывания и компонентов для текущего проекта представлена на рисунке 2.2. Как можно заметить, система состоит из трёх компонентов:

– СУБД MySQL;

– серверное приложение;

– клиентское приложение.

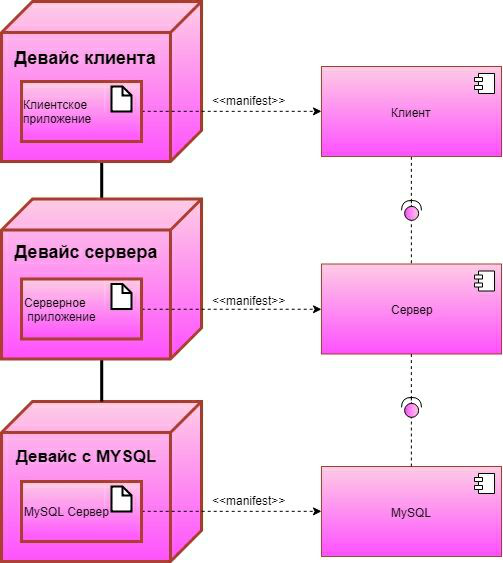


Рисунок 2.2 – Диаграмма развертывания и компонентов

Каждый аспект отвечает за свой функционал. СУБД *MySQL* – за хранение и выборку данных, серверное приложение – за логику системы, а клиентское приложение – за отображение информации.

Диаграмма последовательности – это диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание – деятельность – уничтожение некой сущности) и взаимодействие актеров (действующих лиц) информационной системы в рамках какого-либо определённого прецедента (отправка запросов и получение ответов).

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

На диаграмме последовательности изображен один актёр – пользователь системы, а также следующие объекты:

– клиентское приложение;

– серверное приложение;

– сервер MySQL.

Представлена последовательность действия в три этапа:

1 Пользователь включает приложение, а сервер отправляет клиенту доступ к БД.

2 Клиент запрашивает просмотр таблицы. Приложение отправляет запрос sort, а сервер посылает запрос на сервер MySQL, который выбирает данные и посылает их серверу. Сервер их преобразует и возвращает клиентскому приложению. Приложение отображает информацию для клиента.

3 Клиент нажимает кнопку закрытия в приложении, закрывается GUI, закрывается соединение. Работа приложения завершена.

В этом подразделе произведено обоснование выбора технологий, а также описаны два основных типа диаграммы UML для отображения структуры данных.

## **2.6 Применение паттернов проектирования**

Паттерн проектирования – повторимаяархитектурная конструкция, представляющая собой решение проблемыпроектированияв рамках некоторого часто возникающего контекста, иными словами, паттерны проектирования – это шаблоны необходимых шагов для решения каких-либо определенных задач.

При разработке системы учёта пациентов стоматологической клиники использовались следующие паттерны проектирования: Builder,.

### **2.6.1** Паттерн MVC

Паттерн MVC (Model/View/Controller) состоит из объектов трех видов. Модель – это объект приложения, а вид – экранное представление. Контроллер описывает, как интерфейс реагирует на управляющие воздействия пользователя [8].

Первым с клиентским запросом взаимодействует контроллер, контроллер разбивает данный запрос на элементы, далее он инициализирует объекты модели. После обработки данные отправляются на уровень представления.

Паттерн MVC  – простой способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса.

Диаграмма взаимодействия компонентов для паттерна «MVC» изображена на рисунке 2.3

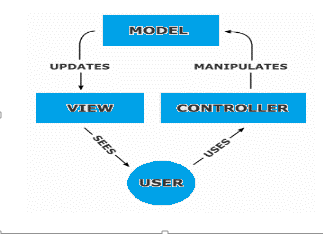


Рисунок 2.3 – Диаграмма для паттерна «MVC»

### **2.6.2** Паттерн Builder

Builder определяет алгоритм поэтапного создания продукта в специальном классе Director (распорядитель), а ответственность за координацию процесса сборки отдельных частей продукта возлагает на иерархию классов Builder. В этой иерархии базовый класс Builder объявляет интерфейсы для построения отдельных частей продукта, а соответствующие подклассы ConcreteBuilder их реализуют подходящим образом, например, создают или получают нужные ресурсы, сохраняют промежуточные результаты, контролируют результаты выполнения операций.

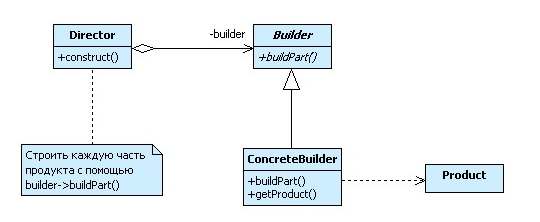


Рисунок 2.4 - UML-диаграмма классов паттерна Builder

### **2.6.3** Паттерн Singleton

Singleton относится к порождающим паттернам. Его дословный перевод – одиночка. Этот паттерн гарантирует, что у класса есть только один объект (один экземпляр класса) и к этому объекту предоставляется глобальная точка доступа. Из описания должно быть понятно, что этот паттерн должен применяться в двух случаях:

1.когда в вашей программе должно быть создано не более одного объекта какого-либо класса. Например, в компьютерной игре у вас есть класс «Персонаж», и у этого класса должен быть только один объект описывающий самого персонажа.

2.когда требуется предоставить глобальную точку доступа к объекту класса. Другими словами, нужно сделать так, чтобы объект вызывался из любого места программы. И, увы, для этого не достаточно просто создать глобальную переменную, ведь она не защищена от записи и кто угодно может изменить значение этой переменной и глобальная точка доступа к объекту будет потеряна. Это свойства Singleton'a нужно, например, когда у вас есть объект класса, который работает с базой данных, и вам нужно чтобы к базе данных был доступ из разных частей программы. А Singleton будет гарантировать, что никакой другой код не заменил созданный ранее экземпляр класса.

**3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

Тестирование является последним этапом разработки программного обеспечения.

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Существуют различные методы тестирования программных средств, такие как:

– функциональное тестирование;

– нагрузочное тестирование;

– стресс-тестирование;

– тестирование стабильности;

– конфигурационное тестирование;

– юзабилити-тестирование;

– тестирование интерфейса пользователя;

– тестирование совместимости.

В данной программе мы используем лишь несколько видов тестирования. А именно:

– Функциональное тестирование;

– Локализация программного обеспечения;

– Интеграционное тестирование.

Для локализации в английском языке иногда применяют сокращение «L10n». Где буквы «L» и «n» – начало и окончание слова Localization, а цифра 10 – количество букв между ними.

Ручное тестирование – часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно производится тестировщиком без использования программных средств, для проверки программы или сайта путём моделирования действий пользователя.

В роли тестировщиков могут выступать и обычные пользователи, сообщая разработчикам о найденных ошибках. Некоторые инструменты для визуального программирования предоставляют возможности и инструменты для облегчения локализации. Например, в [GTK+](https://ru.wikipedia.org/wiki/GTK%2B) чаще всего нет необходимости специально заботиться о разной длине строк в разных языках, так как виджеты автоматически запрашивают необходимый для себя размер.

Однако в большинстве случаев эти возможности значительно ограничены, что сильно отражается на итоговой стоимости локализации.

В этих инструментах нет средств для работы с переводчиком, нет автоматизированных проверок перевода, да и контролировать перевод приложения среднего размера становится невозможно. Поэтому приступая к локализации стоит задуматься о специализированных средствах, заточенных именно на задачу перевода программного обеспечения.

Средства, например, Passolo, которые позволяют переводить меню и сообщения в программных ресурсах и непосредственно в откомпилированных программах, а также тестировать корректность локализации. Для перевода аудиовизуальных материалов (главным образом фильмов) также используются специализированные средства, например, [Swift](https://ru.wikipedia.org/w/index.php?title=Swift_(браузер)&action=edit&redlink=1), которые объединяют в себе некоторые аспекты памяти переводов, но дополнительно обеспечивают возможность появления субтитров по времени, их форматирования на экране, следования.

Интеграционное тестирование – одна из фаз тестирования программного обеспечения, при которой отдельные программные модули объединяются и тестируются в группе.

Обычно интеграционное тестирование проводится после модульного тестирования и предшествует системному тестированию.

Интеграционное тестирование в качестве входных данных использует модули, над которыми было проведено модульное тестирование, группирует их в более крупные множества, выполняет тесты, определённые в плане тестирования для этих множеств, и представляет их в качестве выходных данных и входных для последующего системного тестирования.

Целью интеграционного тестирования является проверка соответствия проектируемых единиц функциональным, приёмным и требованиям надежности. Тестирование этих проектируемых единиц – объединения, множества или группы модулей – выполняется через их интерфейс, с использованием тестирования «чёрного ящика».

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает. Функциональные требования включают в себя: функциональная пригодность, точность, способность к взаимодействию, соответствие стандартам и правилам, защищённость.

При неограниченном времени и ресурсах хорошее тестирование безопасности взломает любую систему. Задача проектировщика системы — сделать цену проникновения более высокой, чем цена получаемой в результате информации. Проблем выявлено не было, что говорит об успешности тестирования.

Вывод: тестирование успешно пройдено, так как проблем выявлено не было ни при одном виде тестирования. Исходя из этого, можно прийти к выводу, что данная система полностью исправна и в доработке не нуждается.

**4 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА**

В данном разделе будет описано руководство пользователя по поводу работы данной программы.

В первую очередь мы запускаем серверное приложение.

После чего должно выскочить окно, предлагающее ввести номер порта. Вводим порт с номером 9999.

Далее нужно нажать на кнопку «Запустить», чтобы сервер начал работу и ожидал подключения от клиентов.

После этого сервер начнёт прослушивание клиентских соединений.

Далее откроется новое окно, которое хранит информацию о нашем сервере:

– порт, на котором запущен сервер;

– количество подключенных клиентов в данный момент.

Следующим шагом мы запускаем клиентскую часть данного приложения, где необходимо будет ввести имя пользователя и пароль. Данная возможность позволяет работать параллельно на различных портах.

Чтобы войти в аккаунт компании, необходимо ввести логин и пароль компании. Данное окно представлено на рисунке 4.4.

Если у сотрудника компании нет аккаунта, есть возможность зарегистрировать данный аккаунт. Тут помимо полей логина и пароля, есть поле бюджет и валюта.

После того как авторизация пройдет успешно, всплывёт окно на стороне клиента, в котором нам на выбор будет предложено три варианта. А конкретно:

– статистика бюджета;

– транзакции компании;

– выход.

А также информация о том, какой тип аккаунта, кто зашел и какой у этого пользователя бюджет.

Нажав на кнопку «Статистика бюджета» откроется график бюджета компании.

Нажав на кнопку «Транзакции компании», нам будет доступен следующий перечень возможностей:

– добавление транзакции расхода;

– добавление транзакции дохода;

– статистика;

– удаление.

Окно с таблицей транзакций представлено на рисунке 4.8.

Нажав кнопку «Статистика», откроется окно аналитической информации, где показана на графике статистика транзакций. Красным цветом обозначается сотрудник, синим – директор.

Теперь рассмотрим вход в систему в качестве администратора. После того как авторизация пройдет успешно, всплывёт окно на стороне клиента, в котором нам на выбор будет предложено восемь вариантов. А конкретно:

– статистика бюджета;

– транзакции компании;

– транзакции сотрудников;

– транзакции руководителей;

– очистить историю транзакций сотрудников;

– очистить историю транзакций руководителей;

– управление аккаунтами;

– выход.

А также информация о том, какой тип аккаунта, кто зашел и какой у этого пользователя бюджет.

Окно представлено на рисунке 4.10.

Нажав на кнопку «Управление аккаунтами», нам будет доступен следующий перечень возможностей:

– заблокировать / разблокировать пользователя;

– сменить пользователю пароль;

– сменить пароль компании;

– сменить валюту компании;

– создать пользователя;

– удалить пользователя.

**ЗАКЛЮЧЕНИЕ**

На этом этапе можно считать завершёнными описание системы, её успешное тестирование и разработку.

Была успешно разработана система Интернет-банкинга и программная поддержка дистанционного обслуживания клиентов.

Для реализации данной задачи использовалось несколько различных технологий, связанных с языком программирования Java.

Например, Socket, Swing, GSON, JFreeChart, MySQL connection и много других технологий.

Для того чтобы протестировать наше программное обеспечение мы использовали три вида тестирования. При реализации каждого из них никаких ошибок выявлено не было. Что, собственно, в какой-то мере, говорит о качестве программы.

Также нами была успешно создана система по управлению базами данных на основе MySQL сервера.

Графический интерфейс в нашем приложении является интуитивно понятным и удобным для использования.

Данная система очень успешно спроектирована. После чего она была отлажена и успешно протестирована.

Описание данной системы можно считать полностью завершённой.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1]JavaRush [Электронный ресурс]. – Режим доступа http:// www.javarush.ru/samouchitel.html.

[2]Программирование на языке Java [Электронный ресурс]. – Режим доступа https://metanit.com/java.

[3]Java Learning [Электронный ресурс]. – Режим доступа – программирование на языке Java .pdf.

[4]Полное руководство по обучению программирования на Java | Java [Электронный ресурс]. – Режим доступа: http://www.internet-technologies.ru/articles/kak-nauchitsya-programmirovat-na-java.html.

[5]Java SoloLearn [Мобильное приложение] – Режим доступа Play Market

[6]MySQL SoloLearn [Мобильное приложение] – Режим доступа Play Market.

**ПРИЛОЖЕНИЕ А (обязательное)**

**Диаграмма классов системы интренет-банкинга.**

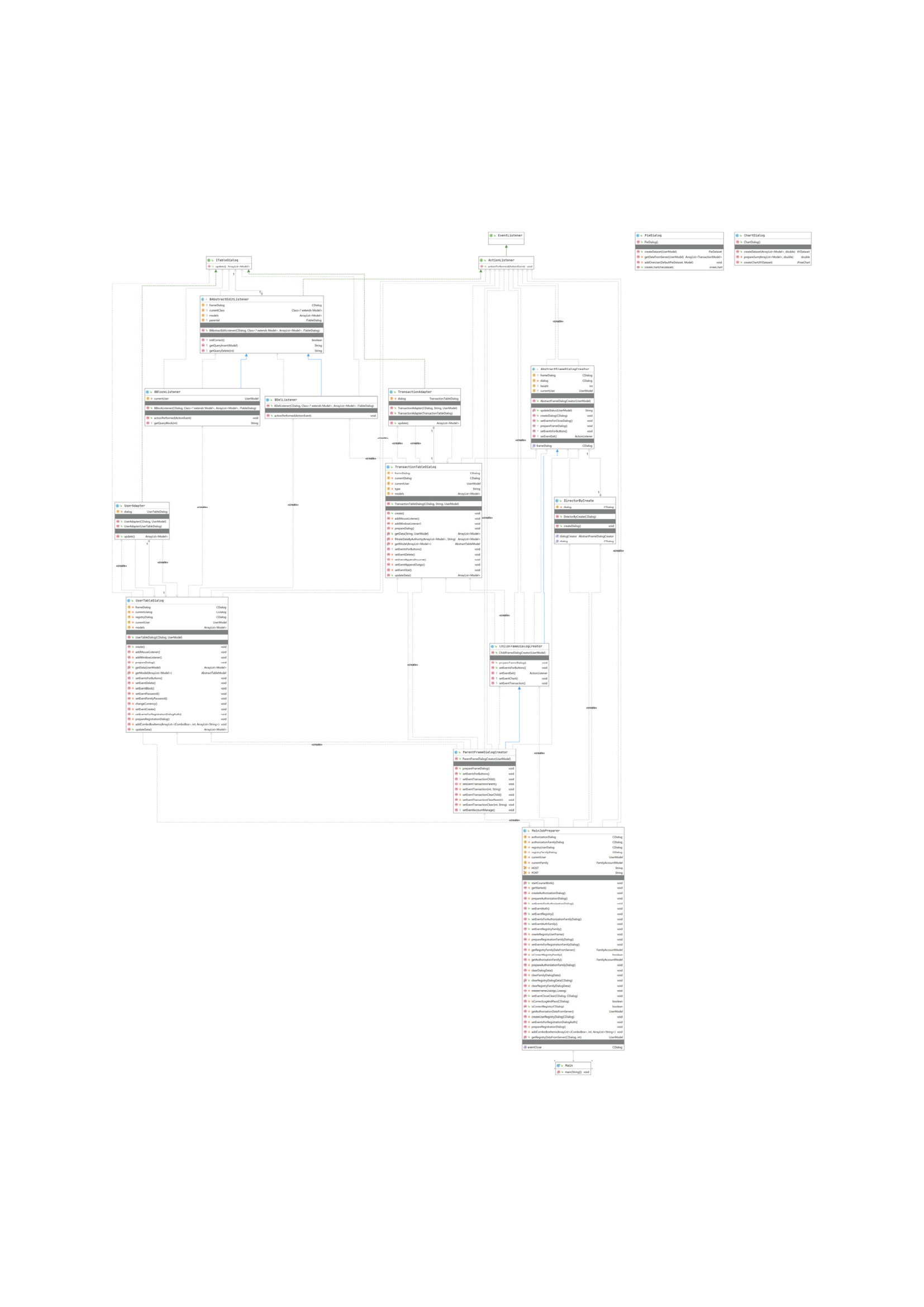


Рисунок Д.1 – Диаграмма классов

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Листинг кода алгоритмов, реализующих бизнес-логику**

package data\_base;

import model.\*;

import validation.Validator;

import java.sql.\*;

import java.util.ArrayList;

// TODO Класс клиентского соединения с БД, паттерн Singleton.

public class ConnectorToDB {

// Приватное свойство, содержит единственный экземпляр класса

private static ConnectorToDB ourInstance = new ConnectorToDB();

private String userName;

private String password;

private String connectionURL;

public String lastError = "";

// Метод для получения объекта (единственного) данного класса

public static ConnectorToDB getInstance() {

return ourInstance;

}

// Приватный конструктор, ограничение создания объектов из вне

private ConnectorToDB() {

lastError = "";

try {

userName = "root";

password = "root";

connectionURL = "jdbc:mysql://localhost:3306/courses?useLegacyDatetimeCode=false&serverTimezone=UTC&useSSL=false";

Class.forName("com.mysql.cj.jdbc.Driver");

} catch (ClassNotFoundException e) {

lastError = "Ошибка соединения с БД: \"" + e.getClass() + "\"";

}

}

private boolean queryNotReturn(String query, String[] params, boolean[] isString){

try(

Connection connection = DriverManager.getConnection(connectionURL, userName, password);

PreparedStatement statement = connection.prepareStatement(query)

){

String y = "";

for (int i = 0; i < params.length; i++) {

y += params[i] + ",-";

}

for (int i = 0; i < params.length; i++) {

if(!isString[i]){

setSpecial(statement, i+1, params[i]);

} else {

if(params[i].equals("NULL"))

statement.setNull(i+1, java.sql.Types.NULL);

else

statement.setString(i+1, params[i]);

}

}

statement.executeUpdate();

statement.close();

connection.close();

return true;

} catch (Exception e){

e.printStackTrace();

lastError = "Ошибка соединения с БД: \"" + e.getClass() + "\"";

return false;

}

}

private void setSpecial(PreparedStatement statement, int id, String data) throws SQLException {

if(!Validator.isInt(data)) {

if (Validator.isDouble(data))

statement.setDouble(id, Double.valueOf(data));

} else statement.setInt(id, Integer.valueOf(data));

}

public boolean insert (String table, String columns, String values, String[] arrayData, boolean[] isStr){

lastError = "";

String query = "insert into `" + table + "`(" + columns + ") values (" + values + ")";

return queryNotReturn(query, arrayData, isStr);

}

public boolean update (String table, int id, String values, String[] arrayData, boolean[] isStr){

lastError = "";

String query = "update `" + table + "` set " + values + " where id" + table + " = ?";

return queryNotReturn(query, getAllStr(arrayData, id), getAllBool(isStr, false));

}

private String[] getAllStr(String[] arrayData, int id){

String[] array = new String[arrayData.length+1];

for (int i = 0; i < arrayData.length; i++) {

array[i] = arrayData[i];

}

array[array.length-1] = String.valueOf(id);

return array;

}

private boolean[] getAllBool(boolean[] arrayData, boolean id){

boolean[] array = new boolean[arrayData.length+1];

for (int i = 0; i < arrayData.length; i++) {

array[i] = arrayData[i];

}

array[array.length-1] = id;

return array;

}

public boolean delete (String table, int id){

lastError = "";

String query = "delete from `" + table + "` where id" + table + " = ?"; //+ id;

return queryNotReturn(query, new String[]{String.valueOf(id)}, new boolean[]{false});

}

private void setSelectParam(PreparedStatement statement, String param, String isInt) throws SQLException{

if(isInt.equals("true")){

if(!Validator.isInt(param)) statement.setDouble( 1, Double.valueOf(param));

else statement.setInt( 1, Integer.valueOf(param));

} else {

statement.setString(1, param);

}

}

public ArrayList<Model> select(String table , String where, String orderBy, String param, String isInt) {

lastError = "";

if(!("".equals(orderBy.trim()))) orderBy = "order by " + orderBy;

if(!("".equals(where.trim()))) where = "where " + where;

String query = "select \* from `" + table + "` " + where + " " + orderBy;

try(

Connection connection = DriverManager.getConnection(connectionURL, userName, password);

PreparedStatement statement = connection.prepareStatement(query)

){

isInt = where.indexOf("LIKE") == -1 ? isInt : "false";

if(!("".equals(where.trim()))) setSelectParam(statement, param, isInt);

ResultSet resultSet = statement.executeQuery();

ArrayList<Model> categories = new ArrayList<>();

while (resultSet.next()) categories.add(resToModel(resultSet, table));

statement.close();

connection.close();

return categories;

} catch (Exception e){

e.printStackTrace();

lastError = "Ошибка соединения с БД: \"" + e.getClass() + "\"";

return null;

}

}

public ArrayList<String> select(String table) {

lastError = "";

String query = "select \* from `" + table + "` ";

try(

Connection connection = DriverManager.getConnection(connectionURL, userName, password);

PreparedStatement statement = connection.prepareStatement(query)

){

ResultSet resultSet = statement.executeQuery();

ArrayList<String> models = new ArrayList<>();

while (resultSet.next()) models.add(resToModelSimple(resultSet, table));

statement.close();

connection.close();

return models;

} catch (Exception e){

lastError = "Ошибка соединения с БД: \"" + e.getClass() + "\"";

return null;

}

}

private Model resToModel(ResultSet resultSet, String table){

try {

switch (table) {

case "transaction\_view": return resToTransactionModel(resultSet);

case "user\_view": return resToUserModel(resultSet);

default: case "familyaccount\_view": return resToFamilyAccountModel(resultSet);

}

} catch (Exception e){

lastError = "Ошибка соединения с БД: \"" + e.getClass() + "\"";

return null;

}

}

private String resToModelSimple(ResultSet resultSet, String table){

try {

switch (table) {

case "authority": return resToAuthorityModel(resultSet);

case "currency": return resToCurrencyModel(resultSet);

default: case "type": return resToTypeModel(resultSet);

}

} catch (Exception e){

lastError = "Ошибка соединения с БД: \"" + e.getClass() + "\"";

return null;

}

}

private String resToAuthorityModel(ResultSet resultSet) throws Exception {

return resultSet.getInt(1) + ") " + resultSet.getString(2);

}

private String resToCurrencyModel(ResultSet resultSet) throws Exception {

return resultSet.getInt(1) + ") " + resultSet.getString(2)

+ ". " + resultSet.getString(3) ;

}

private String resToTypeModel(ResultSet resultSet) throws Exception {

return resultSet.getInt(1) + ") " + resultSet.getString(2)

+ ". " + resultSet.getString(3) ;

}

private TransactionModel resToTransactionModel(ResultSet resultSet) throws Exception {

TransactionModel transaction = new TransactionModel(

resultSet.getInt(1),

resultSet.getString(2),

resultSet.getDouble(3),

resultSet.getString(4),

resultSet.getInt(5),

resultSet.getInt(6),

resultSet.getString(7),

resultSet.getString(8),

resultSet.getInt(9),

resultSet.getString(10),

resultSet.getString(11),

resultSet.getString(12)

);

return transaction;

}

private UserModel resToUserModel(ResultSet resultSet) throws Exception {

UserModel user = new UserModel(

resultSet.getInt(1),

resultSet.getString(2),

resultSet.getString(3),

resultSet.getString(4),

resultSet.getString(5),

resultSet.getString(6),

resultSet.getString(7),

resultSet.getInt(8),

resultSet.getInt(9),

resultSet.getDouble(10),

resultSet.getString(11),

resultSet.getString(12),

resultSet.getInt(13),

resultSet.getString(14),

resultSet.getString(15),

resultSet.getString(16)

);

return user;

}

private FamilyAccountModel resToFamilyAccountModel(ResultSet resultSet) throws Exception {

FamilyAccountModel familyAccount = new FamilyAccountModel(

resultSet.getInt(1),

resultSet.getDouble(2),

resultSet.getString(3),

resultSet.getString(4),

resultSet.getInt(5),

resultSet.getString(6),

resultSet.getString(7)

);

return familyAccount;

}

}

package data\_base;

import model.\*;

import java.util.ArrayList;

public class QueryerToDB {

private ConnectorToDB connector = ConnectorToDB.getInstance();

public String lastError = "";

public ArrayList<TransactionModel> getAllTransactions(String orderBy, String where, String param, String isInt){

try {

ArrayList<Model> models;

ArrayList<TransactionModel> transactions = new ArrayList<>();

models = connector.select(TransactionModel.getViewNameStatic(), where, orderBy, param, isInt);

for (int i = 0; i < models.size(); i++)

transactions.add((TransactionModel) models.get(i));

return transactions;

} catch (Exception e){

lastError = "Ошибка запроса: \"" + e.getClass() + "\"" + connector.lastError;

return null;

}

}

public ArrayList<UserModel> getAllUsers(String orderBy, String where, String param, String isInt){

try {

ArrayList<Model> models;

ArrayList<UserModel> users = new ArrayList<>();

models = connector.select(UserModel.getViewNameStatic(), where, orderBy, param, isInt);

for (int i = 0; i < models.size(); i++)

users.add((UserModel) models.get(i));

return users;

} catch (Exception e){

lastError = "Ошибка запроса: \"" + e.getClass() + "\"" + connector.lastError;

return null;

}

}

public ArrayList<FamilyAccountModel> getAllFamilyAccounts(String orderBy, String where, String param, String isInt){

try {

ArrayList<Model> models;

ArrayList<FamilyAccountModel> familyAccounts = new ArrayList<>();

models = connector.select(FamilyAccountModel.getViewNameStatic(), where, orderBy, param, isInt);

for (int i = 0; i < models.size(); i++)

familyAccounts.add((FamilyAccountModel) models.get(i));

return familyAccounts;

} catch (Exception e){

lastError = "Ошибка запроса: \"" + e.getClass() + "\"" + connector.lastError;

return null;

}

}

public boolean delete(String table, int id) {

lastError = "";

try {

ConnectorToDB connector = ConnectorToDB.getInstance();

if (connector.delete(table,id)) {

return true;

} else {

throw new Exception(connector.lastError);

}

} catch (Exception e){

lastError = "Ошибка запроса: \"" + e.getClass() + "\"" + connector.lastError;

return false;

}

}

public boolean insert(Model model, String table) {

lastError = "";

try {

FieldsTransformation transformator = new FieldsTransformation();

String values = prepareQueryComponents(model);

String columns = prepareQueryColumns(model);

String[] valueByQuery = transformator.newData(model);

if (connector.insert(table, columns, values, valueByQuery, isStrCurrent(model))) {

return true;

} else {

lastError = connector.lastError;

return false;

}

} catch (Exception e){

e.printStackTrace();

if(e.getMessage().length() > 150) lastError = "Ошибка запроса: \"" + e.getClass() + "\"";

else lastError = "Ошибка запроса: \"" + e.getMessage() + "\"";

return false;

}

}

private String prepareQueryComponents(Model model){

if(model instanceof TransactionModel) return "?,?,?,?,?";

if(model instanceof UserModel) return "?,?,?,?,?,?,?,?";

if(model instanceof FamilyAccountModel) return "?,?,?,?";

return null;

}

private String prepareQueryColumns(Model model){

if(model instanceof TransactionModel) return prepareColumns(TransactionModel.COLUMNS);

if(model instanceof UserModel) return prepareColumns(UserModel.COLUMNS);

if(model instanceof FamilyAccountModel) return prepareColumns(FamilyAccountModel.COLUMNS);

return null;

}

private boolean[] isStrCurrent(Model model){

if(model instanceof TransactionModel) return TransactionModel.IS\_STR;

if(model instanceof UserModel) return UserModel.IS\_STR;

if(model instanceof FamilyAccountModel) return FamilyAccountModel.IS\_STR;

return null;

}

private String prepareColumns(String columns[]){

String str = "";

for (int i = 0; i < columns.length; i++) {

str += columns[i];

if(i != columns.length - 1) str +=",";

}

return str;

}

public boolean update(Model model, String table, int id) {

lastError = "";

try {

FieldsTransformation transformator = new FieldsTransformation();

String[] valueByQuery = transformator.newData(model);

String values = prepareQueryValue(model);

if (connector.update(table, id, values, valueByQuery, isStrCurrent(model))) {

return true;

} else {

lastError = connector.lastError;

return false;

}

} catch (Exception e){

if(e.getMessage().length() > 150) lastError = "Ошибка запроса: \"" + e.getClass() + "\"";

else lastError = "Ошибка запроса: \"" + e.getMessage() + "\"";

return false;

}

}

private String prepareQueryValue(Model model){

if(model instanceof UserModel){

return prepareUserValue();

}

if(model instanceof FamilyAccountModel){

return prepareFamilyAccountValue();

}

return "";

}

private String prepareUserValue(){

String str;

str = "firstName = ?, lastName = ?, login = ?, password = ?, phoneNumber = ?, isBlocked = ?, "

+ "idFamilyAccount = ?, idAuthority = ?";

return str;

}

private String prepareFamilyAccountValue(){

String str;

str = "sum = ?, familyLogin = ?, familyPassword = ?, idCurrency = ?";

return str;

}

}

**ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Листинг скрипта генерации базы данных**

CREATE DATABASE IF NOT EXISTS `online\_banking` /\*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;

USE `online\_banking`;

-- MySQL dump 10.13 Distrib 8.0.21, for Win64 (x86\_64)

--

-- Host: 127.0.0.1 Database: online\_banking

-- ------------------------------------------------------

-- Server version 8.0.19

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `authority`

--

DROP TABLE IF EXISTS `authority`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `authority` (

`idAuthority` int NOT NULL AUTO\_INCREMENT,

`authority` varchar(45) NOT NULL,

PRIMARY KEY (`idAuthority`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `authority`

--

LOCK TABLES `authority` WRITE;

/\*!40000 ALTER TABLE `authority` DISABLE KEYS \*/;

INSERT INTO `authority` VALUES (1,'Сотрудник'),(2,'Директор');

/\*!40000 ALTER TABLE `authority` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `currency`

--

DROP TABLE IF EXISTS `currency`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `currency` (

`idCurrency` int NOT NULL AUTO\_INCREMENT,

`currencyCode` varchar(45) NOT NULL,

`currencyName` varchar(45) NOT NULL,

PRIMARY KEY (`idCurrency`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `currency`

--

LOCK TABLES `currency` WRITE;

/\*!40000 ALTER TABLE `currency` DISABLE KEYS \*/;

INSERT INTO `currency` VALUES (1,'RU','Российский рубль'),(2,'BYN','Белорусский рубль'),(3,'USD','Доллар США');

/\*!40000 ALTER TABLE `currency` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `familyaccount`

--

DROP TABLE IF EXISTS `familyaccount`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `familyaccount` (

`idFamilyAccount` int NOT NULL AUTO\_INCREMENT,

`sum` float NOT NULL,

`familyLogin` varchar(45) NOT NULL,

`familyPassword` varchar(255) NOT NULL,

`idCurrency` int NOT NULL,

PRIMARY KEY (`idFamilyAccount`),

KEY `familyAccountIdCurrency\_idx` (`idCurrency`),

CONSTRAINT `familyAccountIdCurrency` FOREIGN KEY (`idCurrency`) REFERENCES `currency` (`idCurrency`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `familyaccount`

--

LOCK TABLES `familyaccount` WRITE;

/\*!40000 ALTER TABLE `familyaccount` DISABLE KEYS \*/;

INSERT INTO `familyaccount` VALUES (1,88107,'Epam','123',1),(2,1131,'MTZ','log',2),(8,12377,'ITechArt','rty',2),(9,888,'BelITSoft','hhh',3),(10,550,'Integral','123',1);

/\*!40000 ALTER TABLE `familyaccount` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Temporary view structure for view `familyaccount\_view`

--

DROP TABLE IF EXISTS `familyaccount\_view`;

/\*!50001 DROP VIEW IF EXISTS `familyaccount\_view`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `familyaccount\_view` AS SELECT

1 AS `idFamilyAccount`,

1 AS `sum`,

1 AS `familyLogin`,

1 AS `familyPassword`,

1 AS `idCurrency`,

1 AS `currencyCode`,

1 AS `currencyName`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `transaction`

--

DROP TABLE IF EXISTS `transaction`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `transaction` (

`idTransaction` int NOT NULL AUTO\_INCREMENT,

`comment` varchar(255) DEFAULT NULL,

`sum` float NOT NULL,

`date` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`idUser` int NOT NULL,

`idType` int NOT NULL,

PRIMARY KEY (`idTransaction`),

KEY `transactionIdUser\_idx` (`idUser`),

KEY `transactionIdType\_idx` (`idType`),

CONSTRAINT `transactionIdType` FOREIGN KEY (`idType`) REFERENCES `type` (`idType`),

CONSTRAINT `transactionIdUser` FOREIGN KEY (`idUser`) REFERENCES `user` (`idUser`)

) ENGINE=InnoDB AUTO\_INCREMENT=46 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `transaction`

--

LOCK TABLES `transaction` WRITE;

/\*!40000 ALTER TABLE `transaction` DISABLE KEYS \*/;

INSERT INTO `transaction` VALUES (1,'Новая операция..',330,'2021-11-03 17:35:29',1,1),(32,NULL,440,'2021-10-02 07:53:37',5,6),(33,NULL,444,'2021-10-02 07:56:13',5,7),(34,NULL,559,'2021-10-02 07:58:15',5,1),(35,NULL,44,'2021-10-02 07:58:39',5,7),(36,NULL,32,'2021-10-02 07:59:08',5,1),(37,NULL,76,'2021-10-02 07:59:29',5,8),(38,'yuiyuyu',660,'2021-10-01 21:12:34',5,1),(41,'',2500,'2021-10-01 22:42:33',3,1),(42,'',5500,'2021-10-01 22:42:47',3,1),(44,'',440,'2021-10-01 22:44:01',3,7),(45,'',1000,'2021-10-01 22:44:15',3,9);

/\*!40000 ALTER TABLE `transaction` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`mysql`@`127.0.0.1`\*/ /\*!50003 TRIGGER `transaction\_AFTER\_INSERT` AFTER INSERT ON `transaction` FOR EACH ROW BEGIN

DECLARE id INT;

DECLARE sumNew float;

DECLARE categoryType VARCHAR(45);

SET categoryType = (SELECT `category` FROM `type` WHERE `type`.`idType` = NEW.`idType`);

IF categoryType LIKE "Расход" THEN

SET sumNew = (-1)\*(NEW.`sum`);

ELSE

SET sumNew = NEW.`sum`;

END IF;

SET id = (SELECT `idFamilyAccount` FROM `user` WHERE `idUser` = NEW.`idUser`);

UPDATE `familyAccount`

SET `familyAccount`.sum = `familyAccount`.sum + sumNew

WHERE `familyAccount`.`idFamilyAccount` = id;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Temporary view structure for view `transaction\_view`

--

DROP TABLE IF EXISTS `transaction\_view`;

/\*!50001 DROP VIEW IF EXISTS `transaction\_view`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `transaction\_view` AS SELECT

1 AS `idTransaction`,

1 AS `comment`,

1 AS `sum`,

1 AS `date`,

1 AS `idUser`,

1 AS `idType`,

1 AS `firstName`,

1 AS `lastName`,

1 AS `idFamilyAccount`,

1 AS `category`,

1 AS `name`,

1 AS `authority`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `type`

--

DROP TABLE IF EXISTS `type`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `type` (

`idType` int NOT NULL AUTO\_INCREMENT,

`category` enum('Расход','Доход') NOT NULL,

`name` varchar(45) NOT NULL,

PRIMARY KEY (`idType`)

) ENGINE=InnoDB AUTO\_INCREMENT=13 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `type`

--

LOCK TABLES `type` WRITE;

/\*!40000 ALTER TABLE `type` DISABLE KEYS \*/;

INSERT INTO `type` VALUES (1,'Доход','Поступление ЗП'),(2,'Доход','Поступление Аванса'),(3,'Доход','Премия'),(4,'Доход','Перевод на карту'),(5,'Расход','Оплата в магазине'),(6,'Расход','Оплата кредита'),(7,'Расход','Оплата в магазине'),(8,'Расход','Оплата в такси'),(9,'Расход','Комисия за уведомления'),(10,'Расход','Перевод на другую карту'),(11,'Расход','Оплата паркинга'),(12,'Расход','Коммунальный платеж');

/\*!40000 ALTER TABLE `type` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `user`

--

DROP TABLE IF EXISTS `user`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user` (

`idUser` int NOT NULL AUTO\_INCREMENT,

`firstName` varchar(45) NOT NULL,

`lastName` varchar(45) NOT NULL,

`login` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

`phoneNumber` varchar(45) NOT NULL,

`isBlocked` varchar(45) NOT NULL DEFAULT 'Активен',

`idFamilyAccount` int NOT NULL,

`idAuthority` int NOT NULL,

PRIMARY KEY (`idUser`),

KEY `userIdFamilyAccount\_idx` (`idFamilyAccount`),

KEY `userIdAuthority\_idx` (`idAuthority`),

CONSTRAINT `userIdAuthority` FOREIGN KEY (`idAuthority`) REFERENCES `authority` (`idAuthority`),

CONSTRAINT `userIdFamilyAccount` FOREIGN KEY (`idFamilyAccount`) REFERENCES `familyaccount` (`idFamilyAccount`)

) ENGINE=InnoDB AUTO\_INCREMENT=14 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `user`

--

LOCK TABLES `user` WRITE;

/\*!40000 ALTER TABLE `user` DISABLE KEYS \*/;

INSERT INTO `user` VALUES (1,'Петров','Петр','petrov','petr','+375 33 808-21-22','Активен',1,1),(2,'Елизавета','Елизина','lisa','lisa','+375 33 808-44-33','Активен',1,2),(3,'Николай','Павловский','pavlo','123','+375 33 808-86-54','Активен',1,2),(5,'Роман','Зайцев','rty','rty','8(029) 302-12-23','Заблокирован',2,2),(11,'Арсений','Романов','ivanj','ivanj','375 33 23782983','Активен',2,1),(12,'asd','fgh','qwe','rty','zxc','Активен',1,2),(13,'ooo','ppp','uuu','iii','kkk','Активен',1,1);

/\*!40000 ALTER TABLE `user` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Temporary view structure for view `user\_view`

--

DROP TABLE IF EXISTS `user\_view`;

/\*!50001 DROP VIEW IF EXISTS `user\_view`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `user\_view` AS SELECT

1 AS `idUser`,

1 AS `firstName`,

1 AS `lastName`,

1 AS `login`,

1 AS `password`,

1 AS `phoneNumber`,

1 AS `isBlocked`,

1 AS `idFamilyAccount`,

1 AS `idAuthority`,

1 AS `sum`,

1 AS `familyLogin`,

1 AS `familyPassword`,

1 AS `idCurrency`,

1 AS `currencyCode`,

1 AS `currencyName`,

1 AS `authority`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Dumping events for database 'courses'

--

--

-- Dumping routines for database 'courses'

--

--

-- Final view structure for view `familyaccount\_view`

--

/\*!50001 DROP VIEW IF EXISTS `familyaccount\_view`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`mysql`@`127.0.0.1` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `familyaccount\_view` AS select `familyaccount`.`idFamilyAccount` AS `idFamilyAccount`,`familyaccount`.`sum` AS `sum`,`familyaccount`.`familyLogin` AS `familyLogin`,`familyaccount`.`familyPassword` AS `familyPassword`,`familyaccount`.`idCurrency` AS `idCurrency`,`currency`.`currencyCode` AS `currencyCode`,`currency`.`currencyName` AS `currencyName` from (`familyaccount` left join `currency` on((`currency`.`idCurrency` = `familyaccount`.`idCurrency`))) \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

--

-- Final view structure for view `transaction\_view`

--

/\*!50001 DROP VIEW IF EXISTS `transaction\_view`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`mysql`@`127.0.0.1` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `transaction\_view` AS select `transaction`.`idTransaction` AS `idTransaction`,`transaction`.`comment` AS `comment`,`transaction`.`sum` AS `sum`,`transaction`.`date` AS `date`,`user`.`idUser` AS `idUser`,`type`.`idType` AS `idType`,`user`.`firstName` AS `firstName`,`user`.`lastName` AS `lastName`,`user`.`idFamilyAccount` AS `idFamilyAccount`,`type`.`category` AS `category`,`type`.`name` AS `name`,`authority`.`authority` AS `authority` from (((`transaction` left join `user` on((`transaction`.`idUser` = `user`.`idUser`))) left join `type` on((`transaction`.`idType` = `type`.`idType`))) left join `authority` on((`user`.`idAuthority` = `authority`.`idAuthority`))) \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

--

-- Final view structure for view `user\_view`

--

/\*!50001 DROP VIEW IF EXISTS `user\_view`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`mysql`@`127.0.0.1` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `user\_view` AS select `user`.`idUser` AS `idUser`,`user`.`firstName` AS `firstName`,`user`.`lastName` AS `lastName`,`user`.`login` AS `login`,`user`.`password` AS `password`,`user`.`phoneNumber` AS `phoneNumber`,`user`.`isBlocked` AS `isBlocked`,`user`.`idFamilyAccount` AS `idFamilyAccount`,`user`.`idAuthority` AS `idAuthority`,`familyaccount`.`sum` AS `sum`,`familyaccount`.`familyLogin` AS `familyLogin`,`familyaccount`.`familyPassword` AS `familyPassword`,`familyaccount`.`idCurrency` AS `idCurrency`,`currency`.`currencyCode` AS `currencyCode`,`currency`.`currencyName` AS `currencyName`,`authority`.`authority` AS `authority` from (((`user` left join `familyaccount` on((`familyaccount`.`idFamilyAccount` = `user`.`idFamilyAccount`))) left join `currency` on((`currency`.`idCurrency` = `familyaccount`.`idCurrency`))) left join `authority` on((`user`.`idAuthority` = `authority`.`idAuthority`))) \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2021-11-16 19:54:19