Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет инженерно-экономический
Кафедра экономической информатики
Дисциплина «Программирование сетевых приложений»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсового проекта
старший преподаватель
_____ Д.А. Сторожев
___.____.2022

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:
«РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ПО РАСЧЁТУ ПРЕМИАЛЬНЫХ ВЫПЛАТ СОТРУДНИКАМ ПРЕДПРИЯТИЯ»

БГУИР КП 1-40 05 01-10 048 ПЗ

Выполнил студент группы 084371
Карпеко Владислав Михайлович

_____
(подпись студента)

Курсовой проект представлен на проверку ___. ____.2022

_____
(подпись студента)

Минск 2022

РЕФЕРАТ

БГУИР КП 1-40 05 01-10 048 ПЗ

Карпеко, В.М. Автоматизация работы предприятий: пояснительная записка к курсовому проекту / В. М. Карпеко. – Минск : БГУИР, 2022. − 50 с.

Пояснительная записка 50 с., 25 рис., 13 источников, 2 приложения

АВТОМАТИЗАЦИЯ РАБОТЫ, ПРЕДПРИЯТИЯ, ФОРМИРОВАНИЕ ДОКУМЕНТОВ, БАЗЫ ДАННЫХ

Цель проектирования: разработка автоматизированной системы по расчёту премиальных выплат сотрудникам предприятия.

Методология проведения работы: в процессе решения поставленных задач были использованы методы компьютерной обработки данных.

Результаты работы: выполнен анализ литературных источников, спроектировано и написано приложение, разработана графическая часть проекта, проведен собственный анализ результатов.

Область применения результатов: проектирование приложений для оптимизации и автоматизации работы предприятий и облегчения работы сотрудников.

СОДЕРЖАНИЕ

Введение 4
1 Описание предметной области 6
1.1 Использование информационных технологий в сфере предприятий . 6
2 Постановка задачи и обзор методов её решения 11
2.1 Постановка задачи расчета премии 11
2.2 Обзор методов решения поставленной задачи 12
3 Функциональное моделирование на основе стандарта IDEF0 14
3.1 Методология IDEF0 14
3.2 Описание разработанной функциональной модели 15
4 Информационная модель системы и её описание 19
5 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы 21
5.1 Схема алгоритма добавления пользователя в базу данных 21
5.2 Схема алгоритма добавления выплаты в базу данных 21
6 Руководство пользователя 24
7 Результаты тестирования разработанной системы 30
Заключение 31
Список использованных источников 32
Приложение А(обязательное) Листинг алгоритмов,реализующих бизнес-логику 33
Приложение Б(обязательное) Листинг основных элементов программы 43

ВВЕДЕНИЕ

Создание современного предприятия – сложный и длительный процесс, который условно можно разделить на два взаимосвязанных этапа. Первый – формирование производственно-управленческой структуры, порождающей мощные потоки информации. Второй – формирование структуры, управляющей этими потоками. Это принято называть комплексной автоматизацией, ведь для того, чтобы работать, необходим инструмент. Это одинаково верно как по отношению к крупным предприятиям, так и по отношению к небольшим организациям – представителям малого бизнеса. Главным объектом автоматизированной системы управления должно быть управленческое решение.

Управленческая деятельность выступает в современных условиях как один из важнейших факторов функционирования и развития организации. Эффективное управление представляет собой ценный ресурс организации, наряду с финансовыми, материальными, человеческими и другими ресурсами. Эта деятельность постоянно совершенствуется в соответствии с объективными требованиями производства и реализации товаров, усложнением хозяйственных связей, повышением роли потребителя в формировании технико-экономических и иных параметров продукции. Следовательно, повышение эффективности управленческой деятельности становится одним из направлений совершенствования деятельности предприятия в целом. Наиболее очевидным способом повышения эффективности протекания трудового процесса является его автоматизация.

Изменения условий производственной деятельности, необходимость адекватного приспособления к ней системы управления, сказываются не только на совершенствовании его организации, но и на перераспределении функций управления по уровням ответственности и формам их взаимодействия. Речь, прежде всего, идет о такой системе управления (принципах, функциях, методах, организационной структуре), которая порождена организационной необходимостью и закономерностью хозяйствования, связанными с удовлетворением индивидуальных потребностей, обеспечением заинтересованности работников в наивысших конечных результатах, растущими доходами населения, регулированием товарно-денежных отношений, широким использованием новейших достижений научно-технической революции. Все это требует от организаций адаптации к новым условиям, преодоления возникающих противоречий в экономическом и научно-техническом процессах.

Наше поколение живет в таком времени, в котором почти все люди владеют той или иной информацией. Поэтому роль информационных технологий огромна в жизни каждого из нас. И, порой невозможно представить нашу жизнь без этих технологий.

На сегодняшний день, с помощью всех современных устройств наша жизнь стала намного проще, а главное – удобнее! Ведь всего пару десятков лет назад человечество и мечтать не могло о том, что может позволить себе сейчас.

Информационные технологии – это процессы, которые используют совокупность средств и методов сбора, обработки и передачи данных (первичной информации) для получения информации нового качества о состоянии объекта, процесса или явления (информационного продукта).

Так же следует отметить появление программ, которые помогают банковским работникам, экономистам, бухгалтерам, проектировщикам в их тяжёлой, повседневной работе. Кроме того, появились детекторы лжи, способные выявлять ложь человека. Это очень помогает в расследовании серьезных преступных дел. Навигационные системы позволяют человеку ориентироваться в том месте, которое он не знает, и всегда проложить маршрут домой. А появление новейших медицинских аппаратов значительно подняло уровень медицины по всему миру - резко понизился показатель смертности, что является огромнейшим плюсом. С помощью них в современном мире человек может вылечиться практически от любой болезни.

Но, к сожалению, не все так идеально как хотелось бы…И у каждого новшества имеются свои недостатки. Так, человечество становится более зависимым от техники.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Использование информационных технологий в сфере предприятий

Передача информации о положении и деятельности предприятия на высший уровень управления и взаимный обмен информацией между всеми взаимными подразделениями фирмы осуществляются на базе современной электронно-вычислительной техники и других технических средствах связи.

Содержание каждой конкретной информации определяется потребностями управленческих звеньев и вырабатываемых управленческих решений. К информации предъявляются определенные требования:

– по объекту и качеству — краткость и четкость формулировок, своевременность поступления;

– по целенаправленности — удовлетворение конкретных потребностей;

– по точности и достоверности — правильный отбор первичных сведений, оптимальность систематизации и непрерывность сбора и обработки сведений.

Крупные компании (корпораций, холдингов), для которых характерна сложная структура, связанная с многопрофильностью подразделений, их территориальной распределенностью и различием в производственном потенциале, как правило, сталкиваются с такими проблемами, как:

– отсутствие организационного единства среди подразделений предприятия, в частности, одинакового понимания сущности бизнес-процессов, единой методологии бухгалтерского учета, унификации нормативно-справочной информации;

– трудности планирования деятельности по всем горизонтам (долгосрочного, текущего, оперативного) на всех уровнях управленческой вертикали, доведения до каждого из подразделений конкретных задач, контроля над текущим исполнением и анализа выполнения этих задач;

– недостаточная оперативность (актуальность) данных о финансово-хозяйственной деятельности подразделений, филиалов и корпорации в целом;

– высокая трудоемкость сбора и обобщения (консолидации) данных территориально-распределенных участков, в частности бухгалтерий, каждая из которых ведет свои, "неполные" с точки зрения корпорации, балансы; большое количество ошибок в подобных данных, их разнородность и несогласованность;

– отсутствие оперативной и достоверной информации о взаиморасчетах (взаимозачетах) с внешними поставщиками и потребителями, а также — филиалами предприятия, и, как следствие, трудность управления дебиторской - кредиторской задолженностью. Решение этой проблемы значительно усложняется при изменении статуса контрагента (например, при покупке корпорацией фирмы, которая ранее была внешним контрагентом).

Единственным решением этих проблем является разработка и внедрение так называемых информационных технологий, т.е. технологий,

основывающихся на использовании вычислительной техники и электронных средств коммуникации.

Согласно определению, принятому ЮНЕСКО, информационная технология - это комплекс взаимосвязанных, научных, технологических, инженерных дисциплин, изучающих методы эффективной организации труда людей, занятых обработкой и хранением информации; вычислительную технику и методы организации и взаимодействия с людьми и производственным оборудованием, их практические приложения, а также связанные со всем этим социальные, экономические и культурные проблемы.

В современном обществе информационные технологии являются универсальным инструментарием в управлении организациями всех типов, действующих во всех сферах. Основные функции современных информационных технологий управления предприятиями – сбор, хранение, поиск, систематизация и обработка необходимых данных для всех сфер общественной жизни, выработка новой информации, решение тех или иных оптимизационных задач. Ставится задача не только отобрать и автоматизировать трудоёмкие, регулярно повторяющиеся рутинные операции над большими массивами данных, но и получить принципиально новую информацию, которая необходима для принятия эффективных управленческих решений.

Среди основных направлений развития современных информационных технологий в обеспечении эффективного функционирования и развития можно выделить:

– автоматизация документооборота;
– коммуникации;
– управление технологией фармацевтического производства;
– автоматизация бухгалтерского учета и планирования;
– разработка систем принятия решений;
– автоматизация банковских операций;
– создание автоматизированных рабочих мест.

Однако при очевидных преимуществах комплексной автоматизации, ее осуществление, тем не менее, связано с рядом проблем. К ним можно отнести необходимость серьезной перестройки существующей модели деловых отношений, сложившейся на предприятии за многие годы, поскольку меняются характер и способы обмена информацией, стиль управления, а также степень влияния и авторитета отдельных людей, оптимизации его организационной структуры, координации работы подразделений, массового обучения сотрудников и т.д. Властные устремления и профессиональный эгоизм мешает принятию решений по существу. К кооперации и интеграции информационно-технологические отделы подготовлены слабо. Часто устанавливаются критерии, которые напрямую не связаны с успехом предприятия. Общие цели ставятся (если это имеет место) на крайне ограниченную временную перспективу. Управление реализацией общих проектов организуется очень плохо. Необходимо четко определить и отразить

в документах функции и обязанности, а также сферу компетенции каждого члена группы специалистов по работе над проектом. Убедиться, что люди, выполняющие эти функции, обладают необходимыми навыками. Разработать подробный план работы, разбить его на этапы, определите сроки выполнения задач и придерживаться их.

В общем случае управление финансами можно представить в виде четырех функциональных уровней рисунок 1.1.

Рисунок 1.1 – Управление финансами

В финансовых подсистемах ERP систем, как правило, предполагается наличие двух способов составления финансового плана: снизу-вверх, сверху-вниз.

В случае использования метода снизу-вверх, соответствующие части финансового плана формируются в низовых подразделениях, после чего система осуществляет их агрегирование. При использовании противоположного метода основные показатели смет определяются на верхнем уровне иерархии предприятия, после чего происходит их детализация на нижних уровнях.

Финансовые планы и бюджеты, количество которых на этапе подготовки, как правило, системой не ограничивается, могут иметь различные версии, модификации и признаки. В качестве рабочего в результате принимается один, который утверждается и объявляется в системе как актуальный.

Все финансовые планы и бюджеты базируются на основе счетов главной книги и заранее описанной в системе управленческой структуры предприятия (центров финансовой ответственности, единиц затрат и др.), определяющей распределение интегрального показателя сметы за период по счету главной книги в соответствии со структурой объектов аналитического (управленческого) учета (центров ответственности, единиц затрат).

Функциональность финансовых подсистем предлагает возможность организации бюджетного контроля и управления движением денежных средств. Как уже упоминалось ранее, бюджетный контроль основывается на единой базе формирования бюджетов и интеграции финансовых операций – Счетах Главной книги и аналитических объектах управленческого учета. Прогнозные данные финансового плана, разбитые по периодам, могут оперативно сравниваться с текущими результатами на счетах главной книги для принятия управленческих решений.

На основе бюджетных данных по аналитическим объектам управленческого учета имеется возможность сравнивать планируемые и фактические результаты по соответствующим статьям затрат/доходов для центров финансовой ответственности. Подсистема финансового плана совместно с подсистемой управления распределением затрат позволяют оценить сходимость результатов плановой и фактической себестоимости выпускаемой продукции, осуществить последующий анализ отклонений, на основе объективных данных сформировать мнение о рентабельности выпускаемой продукции для предприятия и т.д.

Управление движением денежных средств (ДДС), как основная задача казначейства или финансового управляющего, реализуется в системе для планирования и контроля входящих и исходящих денежных потоков рисунок 1.2 и формализации процедур ведения расчетов.

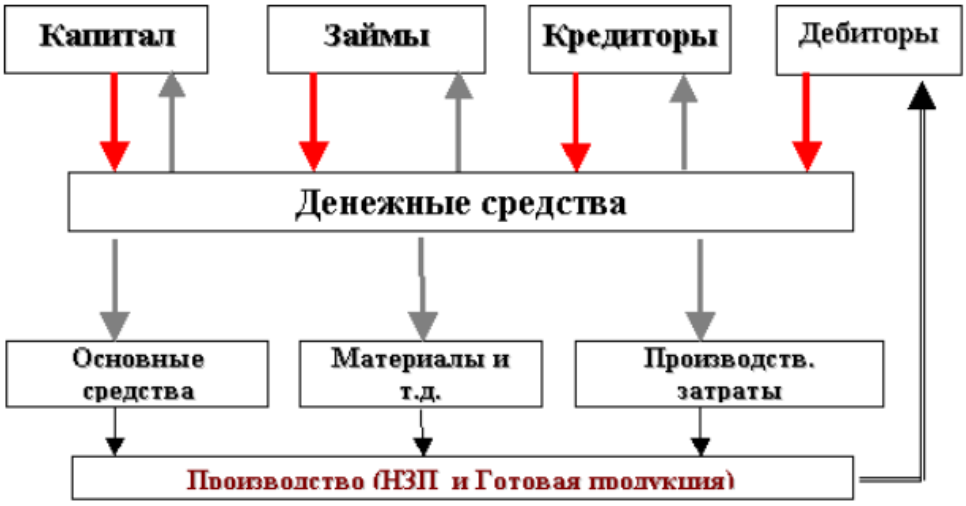


Рисунок 1.2 - Упрощенная схема движения денежных средств

Статус «неразнесенная операция» определяет возможность ее исправления и удаления без каких-либо последствий. Операция с данным статусом еще не является проводкой главной книги и ждет подтверждения корректности и разноски. Процедура контроля неразнесенных операций и их разноски в главную книгу, как правило, выполняется периодически соответствующими должностными лицами по участкам учета. Принимая во внимание интегрированный характер ERP систем, следует заметить, что львиная доля операций порождаются автоматически на основе регистрации первичных документов в подсистемах, связанных с планированием и управлением снабжением, производством, сбытом, проектными работами и т.д. Основная нагрузка по непосредственному ведению операций, как правило, падает на службу расчетов, остальные службы по участкам бухгалтерского учета в большей степени контролируют правильность автоматического формирования операций и осуществляют их разноску.

2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ

2.1 Постановка задачи расчета премии

В данном курсовом проекте поставлена цель сократить трудозатраты сотрудников предприятия, которые ведут учёт записей сотрудников и их премий. Для этого необходимо разработать такое приложение, которое предоставит возможность многим пользователям вносить и просматривать необходимую информацию. Пользоваться им смогут различные сотрудники предприятия.

Предлагается разработать приложение в архитектуре клиент-сервер. Так как необходимость воспользоваться приложением может возникать сразу у нескольких пользователей, сервер должен поддерживать возможность одновременно обрабатывать запросы этих пользователей. Реализация серверного приложения будет осуществлена в виде spring boot приложения на основе RMI общения с клиентом.

Для реализации данного проекта необходимо выполнить следующие задачи:

– изучить предметную область;

– реализовать клиент-серверное приложение;

– настроить базу данных MySQL;

– связать базу данных с сервером;

– создать удобный интерфейс для пользователя;

– разработать собственную иерархию классов;

– реализовать не менее двух паттернов проектирования;

– использовать сокрытие данных, перегрузку методов, переопределение методов, сериализацию, абстрактные типы данных, статические методы;

– предусмотреть обработку исключительных ситуаций;

– предоставить пользователю аналитическую информацию в виде диаграмм;

– реализовать возможность сохранять информацию в файл;

– протестировать полученное приложение.

Для максимального удобства клиентское приложение должно быть реализовано в виде оконного приложения с графическим интерфейсом пользователя. Функционал администратора отличается от функционала простого пользователя, поэтому это будет отражено в клиентском приложении. Администратору будут доступны все функции для работы со списками сотрудников. Простой пользователь сможет просматривать всю информацию, но не вносить изменения.

2.2 Обзор методов решения поставленной задачи

Для программной реализации системы в данном курсовом проекте был выбран объектно-ориентированный язык программирования Java.

Основные преимущества Java как языка программирования:

- является объектно-ориентированным;
- платформонезависимость;
- простота;
- безопасность;
- портативность;
- интерпретируемость;
- высокая производительность.

Ещё одним неоспоримым преимуществом языка Java является его многопоточность. Эта функция позволяет писать программы, которые могут выполнять множество задач одновременно. Применение этой конструктивной особенности позволит реализовать все поставленные требования к функционалу системы.

Графический интерфейс клиентской части реализуется с помощью платформы JavaFX. Она предоставляет большие возможности по сравнению с рядом других подобных платформ, в частности, по сравнению со Swing. Это и большой набор элементов управления, и возможности по работе с мультимедиа, двухмерной и трехмерной графикой, декларативный способ описания интерфейса с помощью языка разметки FXML, возможность стилизации интерфейса с помощью CSS, интеграция со Swing и многое другое.

Для визуального редактирования файлов пользовательского интерфейса будет использоваться редактор SceneBuilder. Интерфейс создаётся с помощью файлов разметки FXML. Этот способ хорошо подходит для отделения контроллеров от представлений, а использование SceneBuilder позволит избежать прямой работы с XML.

Соединение между серверной и клиентскими частями должно устанавливаться с помощью RMI. Это программный интерфейс вызова удаленных методов в языке Java.

При доступе к объекту на другом компьютере возможно вызывать методы этого объекта. Необходимо только передать параметры метода на другой компьютер, сообщить объекту о необходимости выполнения метода, а затем получить обратно возвращаемое значение. Механизм RMI даёт возможность организовать выполнение всех этих операций.

При вызове метода удалённого объекта на самом деле вызывается обычный метод языка Java, инкапсулированный в специальном объекте-заглушке (stub), который является представителем серверного объекта. Заглушка находится на клиентском компьютере, а не на сервере. Она упаковывает параметры удалённого метода в блок байтов. Каждый параметр кодируется с помощью алгоритма, обеспечивающего независимость от аппаратуры. Например, числа всегда передаются в порядке, при котором

сначала передаётся старший байт (big-endian). При этом объекты подвергаются сериализации. Процесс кодирования параметров называется развертыванием параметров (parameter marshaling). Основная цель развёртывания параметров — преобразование их в формат, пригодный для передачи параметров от одной виртуальной машины к другой.

Для хранения информации используется система управления реляционными базами данных MySQL. В реляционной базе данных информация хранится в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. Для упрощения работы с сервером MySQL используется MySQL Workbench. Он представляет графический клиент для работы с сервером, через который в удобном виде можно создавать, удалять, изменять необходимые таблицы и управлять их наполнением.

Для удобной работы с Java используется IntelliJ IDEA. Эта программа имеет множество преимуществ:

– по ходу написания кода программа анализирует его и предлагает, как может код быть дополнен, решены существующие ошибки;

– поддержка широкого круга фреймворков и платформ;

– интеграция с системами контроля версий, к примеру, Git, Subversion, Mercurial, CSV;

– инструменты для работы с базами данных;

– инструменты запуска тестов и анализа покрытия кода и другие.

Также используются такие шаблоны проектирования, как DTO, Singleton, Proxy и другие.

Data Transfer Object (DTO) — один из шаблонов проектирования, используется для передачи данных между подсистемами приложения. Data Transfer Object, в отличие от business object или data access object не должен содержать какого-либо поведения.

Singleton — гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.

Proxy — Позволяет подставлять вместо реальных объектов специальные объекты-заменители. Эти объекты перехватывают вызовы к оригинальному объекту, позволяя сделать что-то до или после передачи вызова оригиналу.

Паттерны делают программу более гибкой, поддерживаемой и устойчивой к изменениям.

Для осуществления обработки исключений используются регулярные выражения, а также конструкция try-catch.

Описанные технологии и программные средства характеризуются в первую очередь надёжностью и простотой использования. Совместное их применение позволит создать удобное многофункциональное приложение, отвечающее всем поставленным требованиям.

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

3.1 Методология IDEF0

Методология функционального моделирования IDEFO – это технология описания системы в целом как множества взаимозависимых действий, или функций. IDEF0 используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции.

Основными понятиями методологии и языка IDEF0 являются блок, ветвление, внутренняя, входная, выходная, граничная стрелки, декомпозиция, дерево узлов.

Блок – это прямоугольник, который содержит имя, номер. Он используется для описания функции.

Ветвление – это разделение стрелки на два и большее число сегментов.

Внутренняя стрелка – это входная, выходная или управляющая стрелка, концы которой связывают источник и потребителя, являющиеся блоками одной диаграммы.

Входная стрелка – стрелка, которая отображает вход IDEF0-блока, то есть данные, которые будут преобразованы функцией в выход. Они располагаются с левой стороны блока.

Выходная стрелка – стрелка, которая отображает выход IDEF0-блока, то есть данные, преобразованные функцией. Они располагаются с правой стороны блока.

Граничная стрелка отображает связь диаграммы с другими блоками системы.

Декомпозиция – разделение моделируемой функции на функции-компоненты.

Дерево узлов – разделение моделируемой функции на функции-компоненты.

Стрелки, входящие в блок сверху - управления. Управления определяют условия, необходимые функции, чтобы произвести правильный выход.

Стрелки, подключенные к нижней стороне блока, представляют механизмы.

При построении IDEF0 имена функций должны быть глаголами или глагольными оборотами.

Наиболее важные свойства объекта выявляются на верхнем уровне иерархии, по мере декомпозиции функции верхнего уровня и разбиения ее на подфункции эти свойства уточняются. Каждая подфункция декомпозируется на элементы следующего уровня. Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования.

3.2 Описание разработанной функциональной модели

Проведённый анализ предметной области даёт возможность разработать функциональную модель процесса автоматизации работы предприятия на основе методологии IDEF0.

Достаточно часто собственники или руководители верхнего уровня ставят задачу разобраться с текущей ситуацией и выяснить целесообразность существующей на предприятии организационной структуры, четко определить функции подразделений, понять, кто за них отвечает, с какой эффективность выполняется работа и т.п. Такая постановка задачи означает, что нужно подвергнуть анализу деятельность всего предприятия, в первую очередь руководителей крупных структурных подразделений. Модель в IDEF0 может помочь в этом случае четко расписать выполняемые функции, структурировать их по подразделениям. Построенную модель можно подвергнуть анализу и предложить изменения как в части структуры подразделений, так и по составу выполняемых ими функций. Модель может быть использована для обсуждения с собственниками и последующего принятия решений по реорганизации компании. При постановке указанной выше задачи никоим образом не говорится о процессном подходе или других технологиях управления. Задача сугубо конкретная. Способы ее решения должны быть максимально просты, и в то же время эффективны. При таком подходе можно строить модель в IDEF0, опираясь на схему организационной структуры предприятия.

Пример организационной структуры компании представлен на рисунке 3.1.

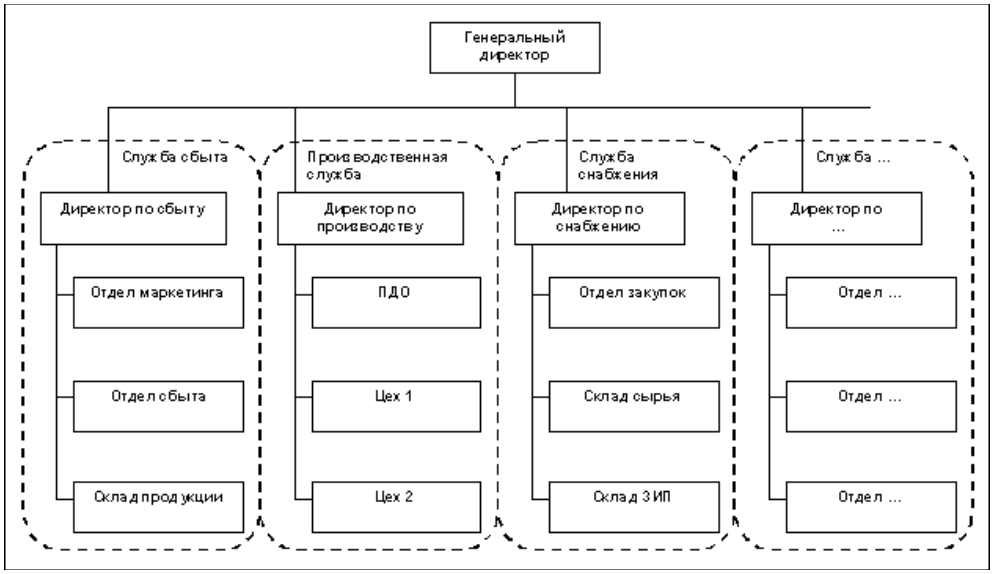
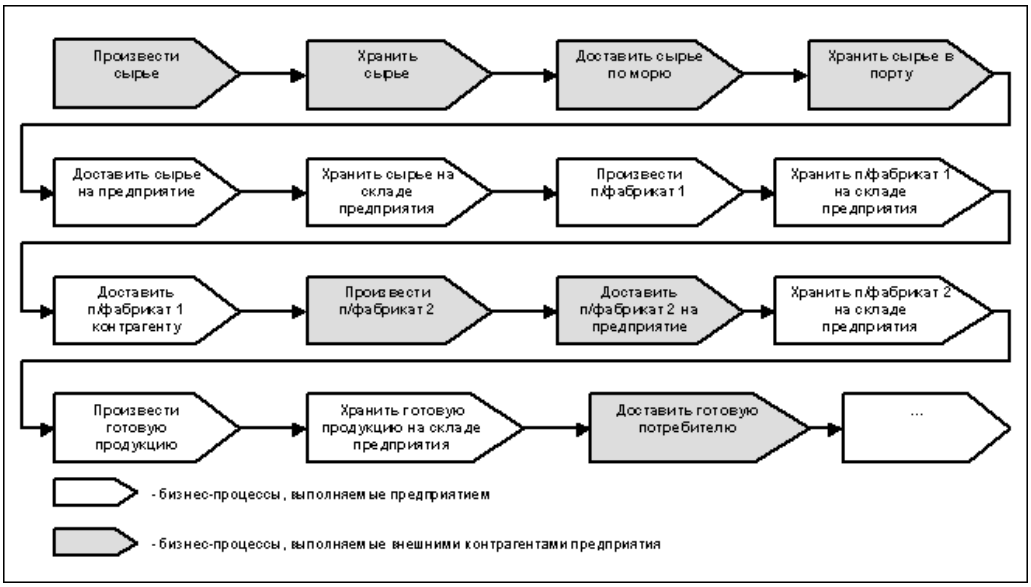


Рисунок 3.1 - Фрагмент организационной структуры компании

Модели в IDEF0, построенные на основе организационной структуры предприятия, хорошо отражают его текущее состояние с точки зрения структуры и выполняемых функций. Однако процессы оказываются выделенными фактически только на уровне небольших отделов — там, где уже нет никаких подразделений, а существует разделение обязанностей между отдельными сотрудниками. Для анализа и реорганизации деятельности предприятия в целом такая степень детализации, как правило, оказывается излишней — чрезмерный объем информации затрудняет принятие решений. Межфункциональные (или «сквозные») бизнес-процессы предприятия при таком подходе оказываются невыделенными. Поэтому, если возникает необходимость анализа и реорганизации предприятия именно с точки зрения оптимизации крупных, межфункциональных бизнес-процессов, необходимо использовать другие подходы, например метод построения моделей в IDEF0 на основе цепочек создания ценности.

Для этого необходимо выявить цепочку последовательно выполняемых бизнес-процессов, на выходе которой появляются продукты (услуги) предприятия. В такую цепочку процессов, как правило, включаются не только процессы предприятия, но и процессы, выполняемые его внешними контрагентами. На рисунке 3.2 представлен фрагмент цепочки создания ценности рассматриваемого предприятия.



Ртсунок 3.2 - Фрагмент цепочки ценности

Первая заключается в использовании данных стандартов и получаемых моделей для управления организацией. Первоначальное использование IDEF, как и введение понятий "бизнес-процесс", "стандарт моделирования деятельности компании", "методология описания" и т. д., в российской компании чаще всего связано с внедрением на предприятии информационной системы. Можно даже сказать, что информационные технологии сейчас в принципе выступают мощным "локомотивом" изменений, который приводит

в движение все остальные части компании. Почему именно информационные технологии? Во-первых, потому что с изменением бизнес-среды перед предприятием встают не только новые оперативные вопросы, но и появляются новые стратегические задачи развития, решение которых требует новую информацию, причем качественно новую, отражающую не только состояние, но и само строение бизнес-системы. Во-вторых, в информационных системах отражаются самые последние технические достижения, а также опыт и знания в предметных областях менеджмента. В-третьих, информационная система объединяет все подразделения компании, позволяя автоматизировать многие функции по сбору и обработке информации. Но вместе с тем, цель автоматизации основных процессов текущей операционной деятельности, которая ставится перед внедрением ИС, и предопределяет результаты аналитиков и разработчиков, сводящиеся к описанию бизнес-процессов на уровне конечных исполнителей, практически без учета деятельности руководителей. Обычно последующее за этим шагом решение непосредственно включить некоторые функции управления в информационную систему и необходимость в каждом процессе видеть место руководителя приходит позднее, когда создание моделей без учета деятельности руководителей входит в привычку. Таким образом, можно сделать первое умозаключение, что само по себе моделирование бизнес-процессов или наличие информационной системы ничего не значат с точки зрения поддержки процесса управления, пока руководителем явно не будет поставлена цель "включить меня в процесс".

Вторая проблема является следствием первой и связана с представлением разработанных моделей менеджерам компании. Дело в том, что полученные схемы и описания наиболее удобны разработчикам информационных систем и аналитикам, но не всегда выразительны и наглядны. Особенно если необходимо сделать презентацию сложного процесса с целью обсуждения возможных его улучшений и сравнить два варианта ("as is" и "to be"). То есть проблема состоит не в какой-либо методологической сложности IDEF и не в том, что менеджеры не способны ее изучить (или не хотят этого делать), а именно в выразительности схем при обсуждении и принятии решений большим количеством участников в сжатые сроки (в объеме презентации).

После перечисленного выше была разработана четырехуровневая функциональная модель процессов предметной области рисунок 3.3.

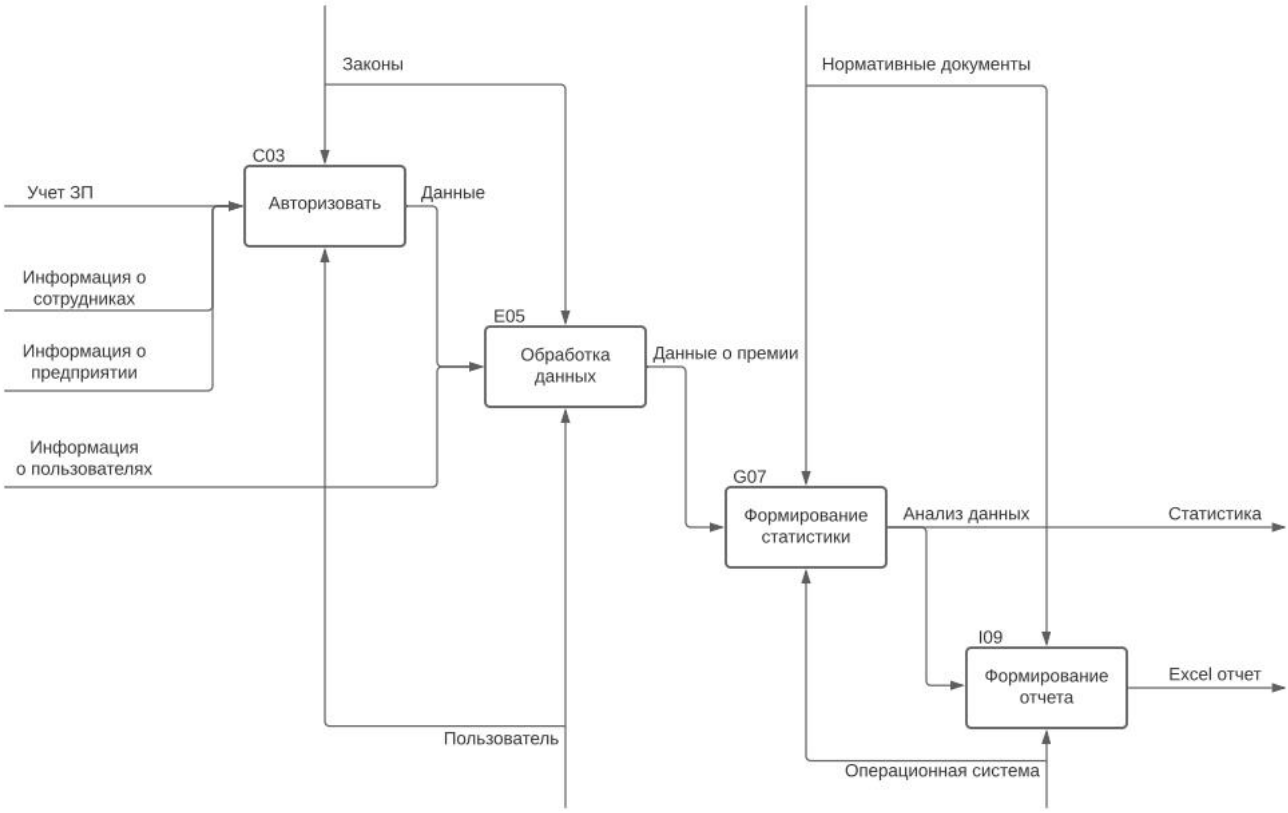


Рисунок 3.3 - Функциональная модель процессов предметной области

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ

Информационная система – это коммуникационная и вычислительная система, которая предназначена для сбора, хранения, обработки и передачи информации. Она также снабжает работников различного ранга той информацией, которая необходима им для реализации функций управления.

Информационная модель системы отражает информацию о предметной области, данные о которой должны храниться в проектируемой базе данных. В данном курсовом проекте предметной областью является автоматизация работы медицинских учреждений.

В процессе информационного моделирования были выделены следующие сущности:

- пользователи;
- отдел;
- выплата.

Сущность «Пользователи» служит для предоставления возможности входа в систему. Наличие такой сущности позволяет разграничить пользователей системы по типу (администратор или простой пользователь) и тем самым предоставить каждому пользователю необходимый ему функционал. Атрибутами этой сущности являются логин, пароль, имя, роль отдел, выплаты и уникальный ID для каждого пользователя.

Пользователь, когда он находится в режиме администратор, имеет полный функционал:

- создание пользователя;
- просмотр пользователей;
- обновление пользователя;
- удаление пользователя;
- создание выплаты;
- просмотр выплат;
- обновление выплаты;
- удаление выплаты;
- формирование статистики;
- просмотр диаграммы.

В свою очередь обычный пользователь может только просматривать данные.

Сущность «Отдел» нужна для хранения информации об отделе, в котором работает пользователь.

Сущность «Выплата» нужна для хранения информации о выплатах, то есть дата выплаты, заработная плата и премия. Данные в эту сущность могут заноситься как при помощи базы данных, так и с помощью администратора.

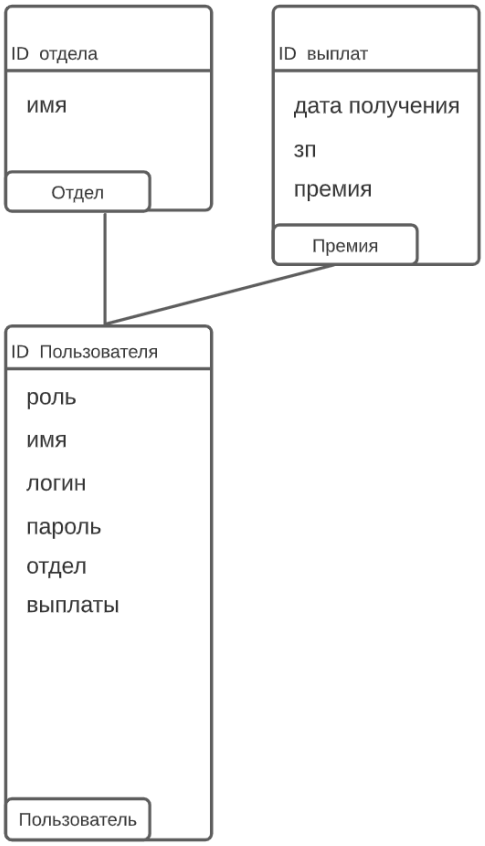


Рисунок 4.1 – Информационная модель базы данных

С учётом обозначенного взаимодействия сущностей смоделируем их взаимодействие в формате IDEF1.X и приведём эту модель к третьей нормальной форме. В результате последовательного приведения получается модель, соответствующая условиям третьей нормальной формы – не ключевой атрибут сущности функционально зависит только от всего первичного ключа и ни от чего другого (рисунок 4.1).

5 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

5.1 Схема алгоритма добавления пользователя в базу данных

Схема алгоритма добавления пользователя в базу данных представлена на рисунке 5.1.

Рисунок 5.1 – Схема алгоритма добавления пользователя в базу данных

Выполнение данного алгоритма начинается с определением значений параметров для вставки в базу данных. Далее происходят процессы создания сущности для сохранения в базу данных. Так как мы используем фреймворк Hibernate для маппинга наших данных из базы, то будут выполнены следующие методы:

- подключение к сессии;
- открытие транзакции;
- сохранение сущности;
- коммит транзакции;
- отключении от сессии;

5.2 Схема алгоритма добавления выплаты в базу данных

Рассмотрим на рисунке 5.2 схему алгоритма добавления выплаты в базу.

Выполнение этого алгоритма начинается с создания нашей сущности выплаты, то есть установки даты выплаты, зарплаты и премии. Далее мы должны получить айди пользователя, к которому будет добавлена данная выплата. После чего происходит работа Hibernate, подключение к сессии, открытие транзакции, поиск пользователя по айди, добавление выплаты, сохранение пользователя, коммит транзакции и отключение от сессии. На этом работа алгоритма завершается.

Стоит упомянуть, почему в качестве ORM был выбран Hibernate.

Hibernate удаляет множество повторяющегося кода из JDBC API, а следовательно, его легче читать, писать и поддерживать.

Hibernate поддерживает наследование, ассоциации и коллекции, что не доступно в JDBC API.

Hibernate неявно использует управление транзакциями. Большинство запросов нельзя выполнить вне транзакции. При использовании JDBC API для управления транзакциями нужно явно использовать commit и rollback.

JDBC API throws SQLException, которое относится к проверяемым исключениям, а значит необходимо постоянно писать множество блоков try-catch. В большинстве случаев это не нужно для каждого вызова JDBC и используется для управления транзакциями. Hibernate оборачивает исключения JDBC через непроверяемые JDBCException или HibernateException, а значит нет необходимости проверять их в коде каждый раз. Встроенная поддержка управления транзакциями в Hibernate убирает блоки try-catch.

Рисунок 5.2 – Схема алгоритма добавления доктора в базу данных

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для корректного запуска системы, компьютер или рабочая станция, на которой разворачиваются клиентская и серверная части приложения, должен обладать следующими минимальными требованиями:

- 32-разрядная система ОС Windows 7 и выше;
- СУБД MySQL, MySQL Workbench;
- Java Development Environment (IntelliJ IDEA Ultimate).

Серверная и клиентская часть программы реализована в разных проектах. Для начала работы требуется сначала запустить проект сервера. После этого в консоли IntelliJ IDEA появится сообщение об удачном запуске сервера, номер порта и IP адрес пользователя (рисунок 6.1).

Рисунок 6.1 – Запуск сервера

На втором этапе запуска программы нужно запустить проект клиента. После запуска проекта клиента появляется окно авторизации, а также в консоли IntelliJ IDEA появится сообщение об удачном подключении к серверу и данные о подключении (рисунок 6.2, 6.3)

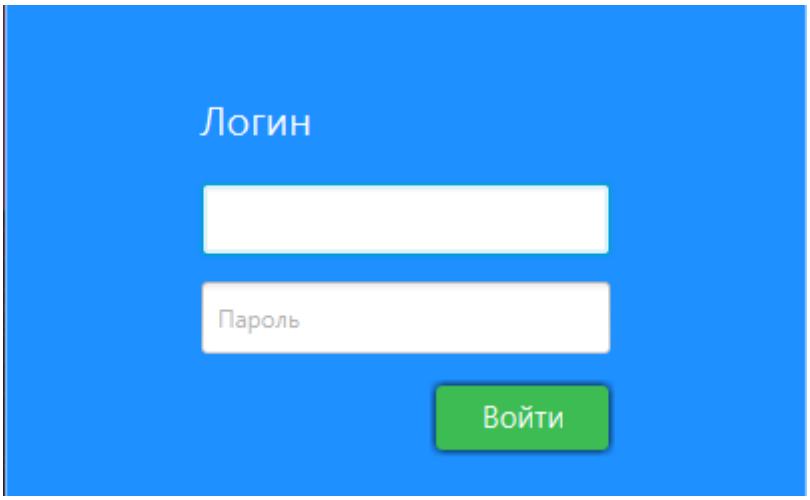


Рисунок 6.2 – Окно авторизации

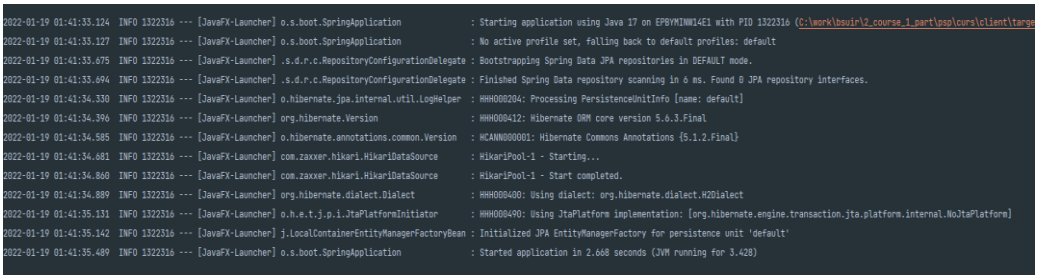


Рисунок 6.3 – Запуск клиента

Далее, в появившемся окне авторизации, предлагается ввести логин пароль. Если пользователь хочет зайти как администратор, то необходимо ввести логин «admin», а если как простой пользователь, то «user».

После ввода правильного логина и пароля появляется уведомление об успешной авторизации (рисунок 6.4).

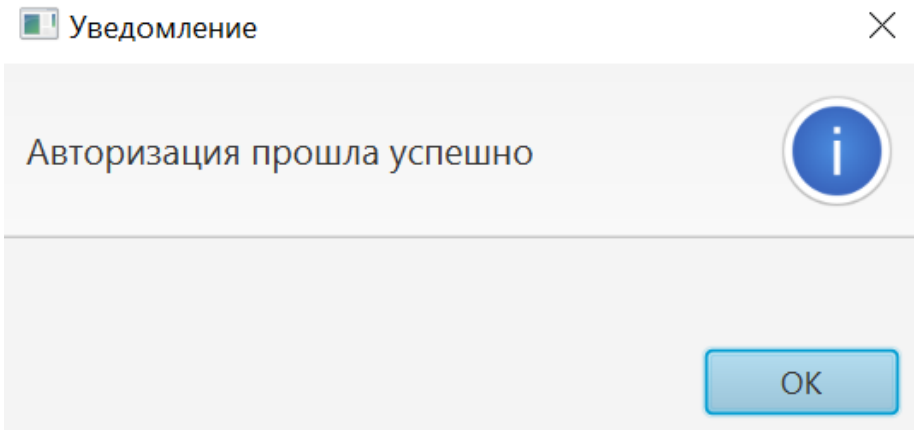


Рисунок 6.4 – Уведомление об успешной авторизации

Затем появляется главное окно. Администратору доступны все функции программы на рисунке 6.5 и рисунок 6.6 режим пользователя.

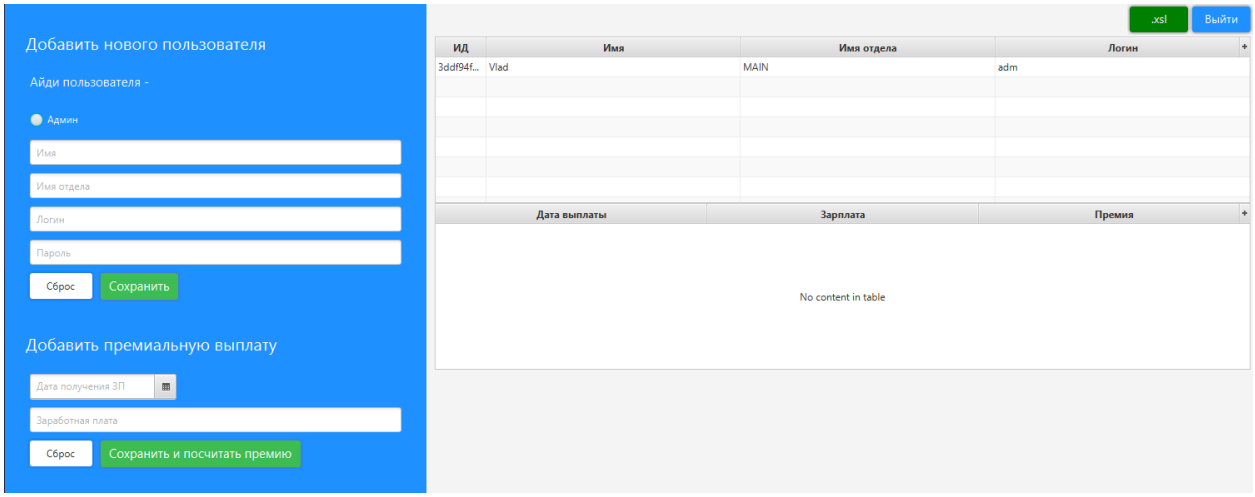


Рисунок 6.5 – Главное окно в режиме администратора

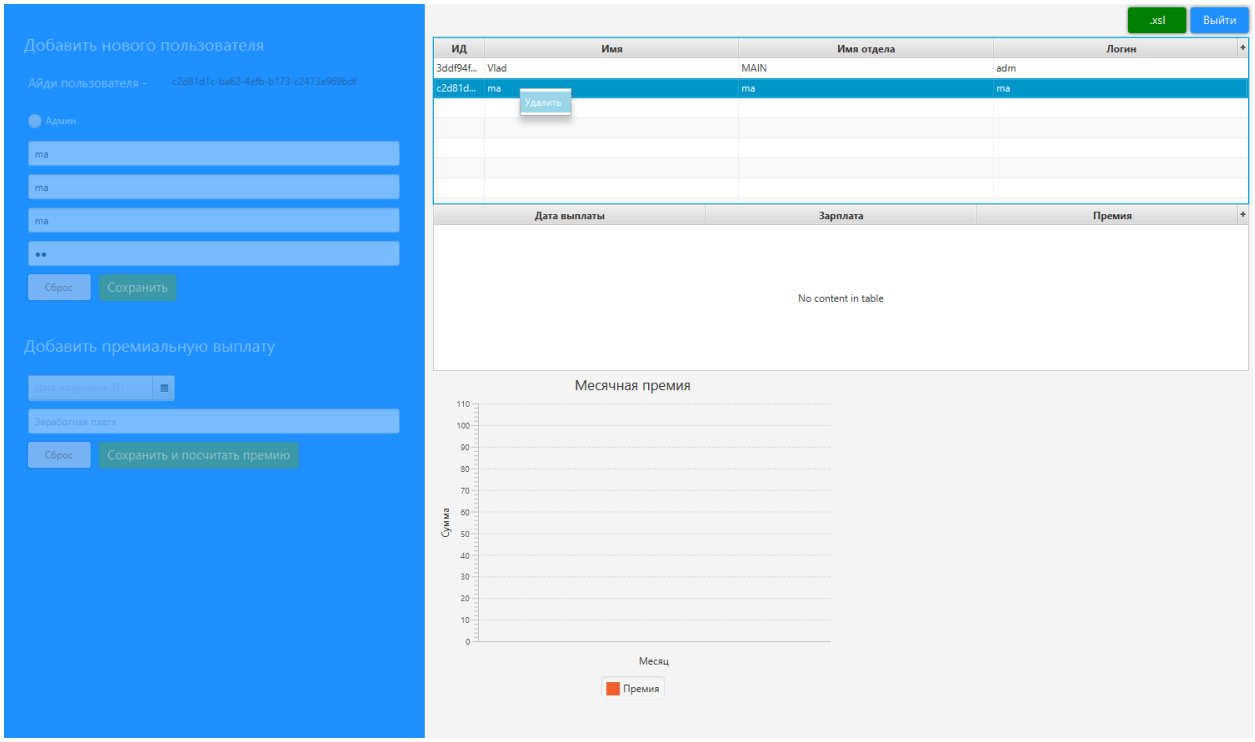


Рисунок 6.6 – Главное окно в режиме пользователя

Для того, чтобы добавить бодьзователя нужно ввести данные в левых верхних колонках и нажать сохранить, но если вдруг, что-то пошло не по плану, то всегда можно сбросить эти данные, рисунок 6.7.

Точно такая же реализация и для добавления выплаты, но приэтом, чтобы связать пользователя и выплату, нужно предварительно выбрать пользователя в табличке пользователя как на рисунке 6.8.

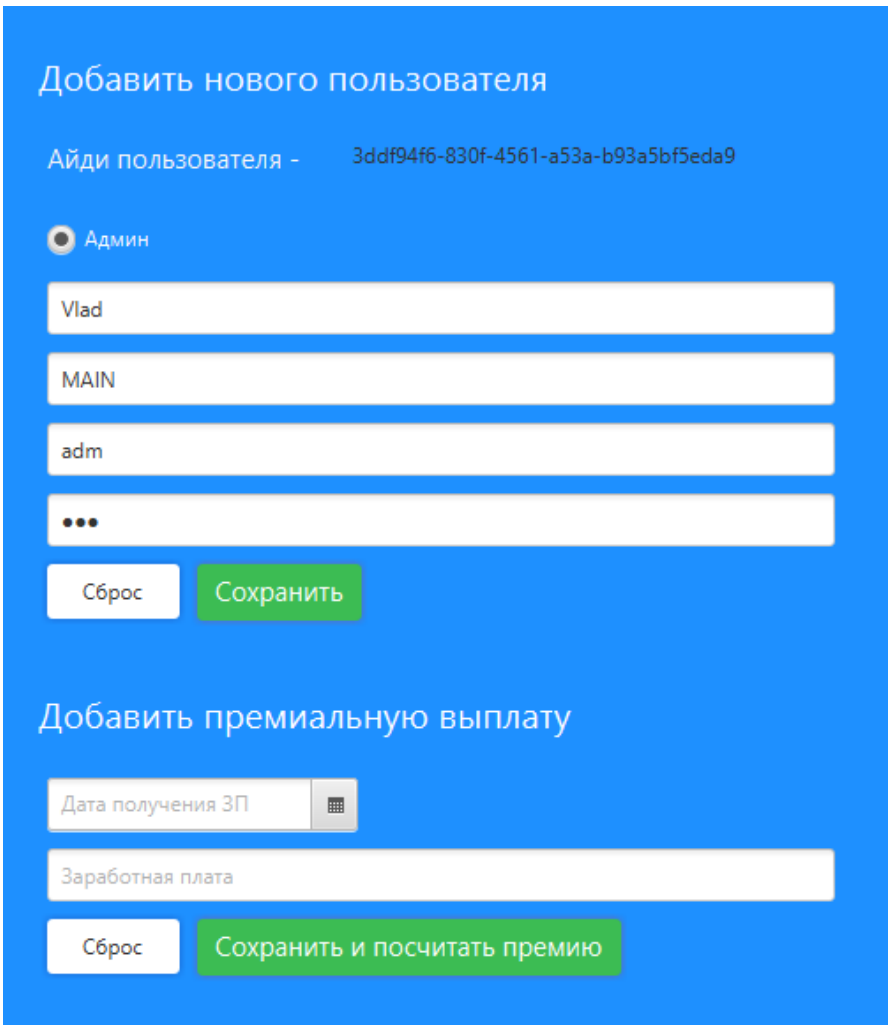


Рисунок 6.7 – Добавление новой записи пользователя

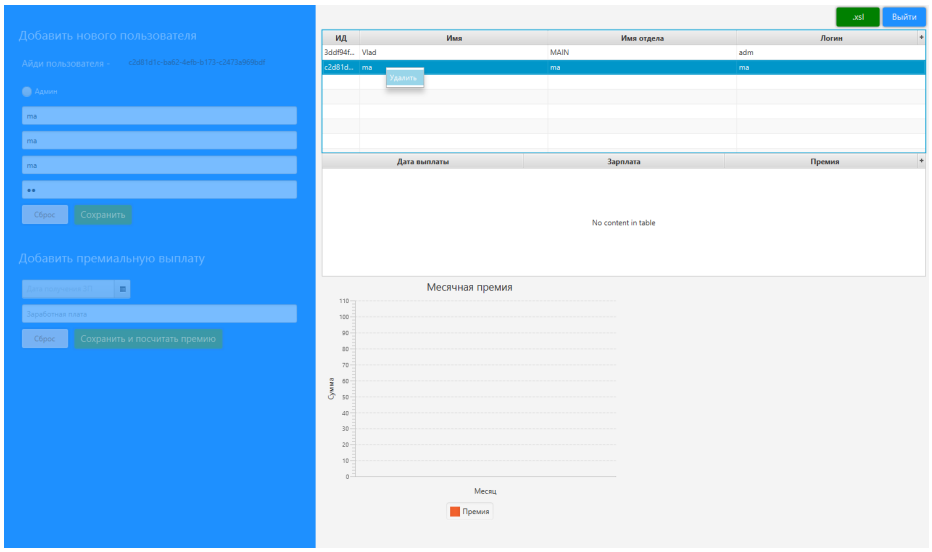


Рисунок 6.8 – Добавление новой записи выплаты

При правильном вводе данных появляется уведомление об успешном добавлении данных и предупреждение о переходе на главную форму (рисунок 6.9).

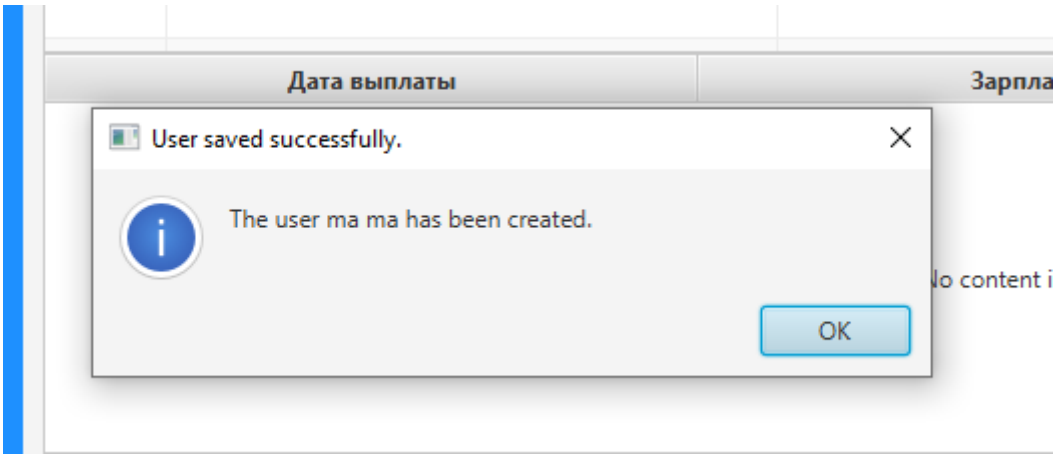


Рисунок 6.9 – Уведомление об успешном добавлении данных

Чтобы удалить запись в таблице, ее нужно выделить и нажать на кнопку «Удалить». После этого появляется уведомление об подтверждении удалении данных (рисунок 6.10).

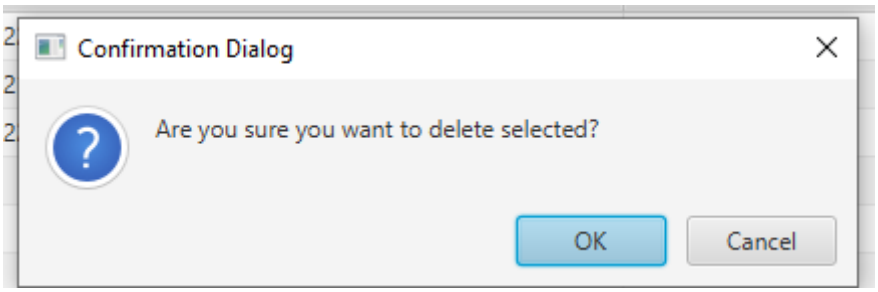


Рисунок 6.10 – Уведомление об подтверждении удалении данных

Также при выборе конкретного пользователя у него появляются все его выплаты в таблице ниже, и на основе этого формируется диаграмма о его размере премии в соотношении к каждому месяцу, когда была начислена премиальная выплата. Это представлено на рисунке 6.11

Рисунок 6.11 – Статистика

На рисунке 6.12 показано диалоговое окно, с помощью которого происходит управление всеми записями в базе данных.

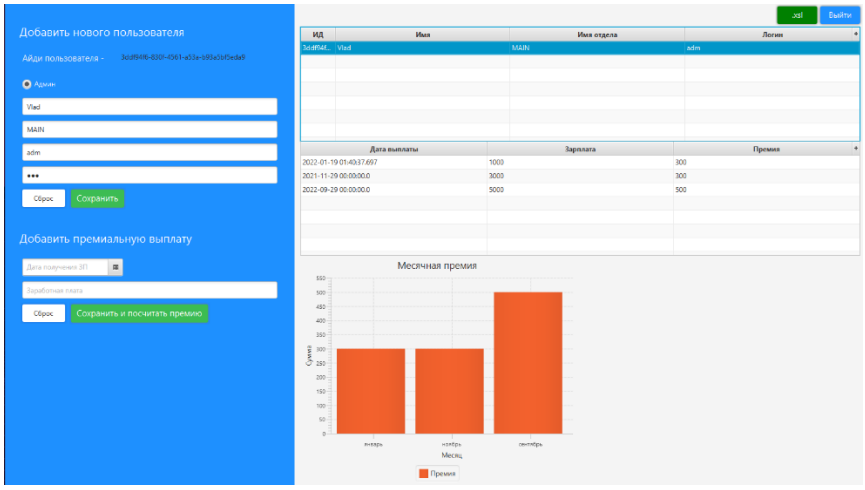


Рисунок 6.12 – Управление базой данных

При нажатии на кнопку «.xls» будет составлен excel отчет обо всех выплатах выбранного пользователя, рисунок 6.13.

	A	B	C	D
1	**Дата получения ЗП**	**ЗП**	**Премия**	
2	19/01/2022	1000	300	
3	29/11/2021	3000	300	
4	29/09/2022	5000	500	
5				
6				
7				
8				
9				

Рисунок 6.13 – Отчет

7 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

В процессе работы данного приложения по разным причинам могут возникать различные ошибки. Для устойчивого функционирования необходимо предусмотреть обработку исключительных ситуаций.

Если в окне авторизации неправильно введен пароль и/или логин, то появляется окно об ошибке, в противном случае – окно об успешной авторизации (рисунок 8.1,8.2).

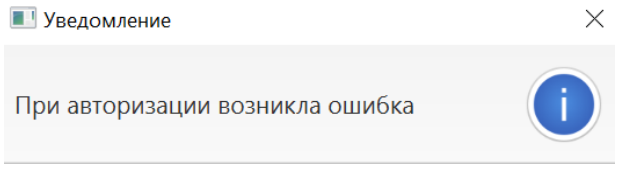


Рисунок 7.1 – Ошибка авторизации

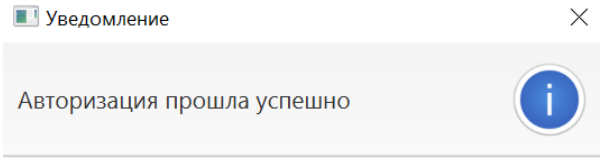


Рисунок 7.2 – Успешная авторизация

При корректном добавлении и удалении данных появляются уведомления говорящие об этом (рисунок 8.3,8.4).

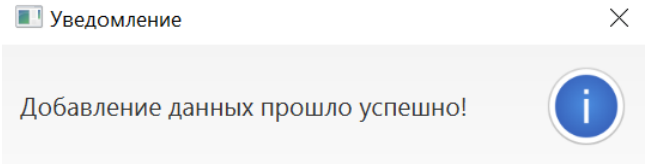


Рисунок 7.3 – Успешное добавление данных

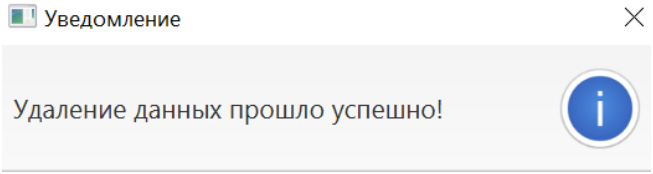


Рисунок 7.4 – Успешное удаление данных

ЗАКЛЮЧЕНИЕ

В результате проделанной работы было создано программное средство, дающее возможность упростить работу сотрудников предприятий для расчета и мониторинга премиальных и заработных выплат.

В ходе создания программного средства были подробно изучены особенности внедрения информационных технологий в сферу предприятий. Полученная информация позволила построить функциональную модель IDEF0, которая наглядно отображает процесс работы предприятий. Были показаны и описаны диаграммы UML, с помощью которых было выполнено проектирование системы. Также была рассмотрена архитектура созданного программного средства. Помимо этого, в ходе выполнения проекта было составлено руководство пользователю, где понятным и доступным языком описывается принцип работы программы. В завершение работы было проведено тестирование разработанной системы, подтвердившее работоспособность созданного программного средства.

Основные функции программного средства реализованы в соответствии с выявленными особенностями предметной области. Был разработан довольно широкий функционал для работы с информацией, которая содержится в базе данных. Стиль интерфейса программы создавался с упором на массовость потребления и использования, который позволит любому пользователю легко и просто использовать данное программное средство без лишних временных затрат.

Бизнес-логика системы даёт возможность систематизировать данные в удобной форме. Полученные результаты могут быть очень эффективны при подведении итогов работы учреждения, а также при определении дальнейшей стратегии развития.

Одним из отличий данного программного средства является надёжная и безопасная база данных, а также структурированная система учёта информации. Благодаря этому вероятность искажения информации минимальна.

В будущем возможно рассмотрение вопроса о расширении функционала программы или же усовершенствования имеющегося. Это обеспечит расширение спектра применения разработанного программного средства.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Беловодский, А. А. Здравоохранение в России: проблемы и пути решения / А. А. Беловодский // Современные наукоёмкие технологии. – 2009. – №11. – С. 21 – 27.

[2] Национальный правовой интернет-портал Республики Беларусь [Электронный ресурс]. – Электронные данные. – Режим доступа: http://pravo.by/document/?guid=3871&p0=v19302435.

[3] Академия профессионального развития [Электронный ресурс]. – Электронные данные. – Режим доступа: https://academy-prof.ru/blog/informacionnye-tehnologii-v-medicine.

[4] ALP Group [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.alp.ru/itsm/interesting/informatsionnyie-tehnologii-v-zdravoohranenii.

[5] Блинов, И. Н. Java. Промышленное программирование: практическое пособие / И. Н. Блинов, В. С. Романчик. – Мн.: Универсал-Пресс, 2012.

[6] Буч, Г. UML. Классика CS / Г. Буч, Дж. Рамбо, А. Якобсон; пер. с англ. – 2-е изд. – СПб.: Питер, 2006.

[7] Методология IDEF0 [Электронный ресурс]. – Режим доступа: https://itteach.ru/bpwin/metodologiya-idef0

[8] Проектирование, разработка и сопровождение баз данных с использованием CASE – средств [Электронный ресурс]. – Режим доступа: https://www.bsuir.by/m/12_100229_1_90142.pdf

[9] Протокол TCP/IP моделирования [Электронный ресурс]. – Режим доступа: http://chernykh.net/content/view/553/751/

[10] YouTube [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.youtube.com/user/PlurrimiTube/featured.

[11] JavaRush [Электронный ресурс]. – Электронные данные. – Режим доступа: https://javarush.ru.

[12] Министерство здравоохранения Республики Беларусь [Электронный ресурс]. – Электронные данные. – Режим доступа: http://minzdrav.gov.by/ru/dlya-belorusskikh-grazhdan/uchrejdenia-zdravoohranenia/polikiliniki.php.

[13] Блог Алексея Гулынина [Электронный ресурс]. – Электронные данные. – Режим доступа: https://alekseygulynin.ru/chat-na-java-servernaya-chast.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг алгоритмов, реализующих бизнес-логику

```
package by.bsuir.psp.client.controller;

import by.bsuir.psp.client.config.StageManager;
import by.bsuir.psp.client.view.FxmlView;
import by.bsuir.psp.model.dto.DepartmentDto;
import by.bsuir.psp.model.dto.PaymentDto;
import by.bsuir.psp.model.dto.UserDto;
import by.bsuir.psp.model.dto.UserRole;
import by.bsuir.psp.model.dto.service.UserService;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.chart.BarChart;
import javafx.scene.chart.CategoryAxis;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ButtonType;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.RadioButton;
import javafx.scene.control.SelectionMode;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.Pane;
import lombok.Getter;
import lombok.Setter;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Lazy;
import org.springframework.stereotype.Controller;

import java.net.URL;
import java.text.SimpleDateFormat;
import java.time.ZoneId;
import java.util.Collections;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Objects;
import java.util.ResourceBundle;
import java.util.UUID;

@Setter
@Getter
@Slf4j
@Controller
public class UserController implements Initializable {

  private static final double AWARD_PERCENT = 0.1;

  @FXML
```

```
private boolean currentUserBasic;

@FXML
private RadioButton rbAdmin;

@FXML
private Label userId;

@FXML
private TextField txName;

@FXML
private TextField txDepartmentName;

@FXML
private TextField txLogin;

@FXML
private PasswordField txPassword;

@FXML
private DatePicker txPaymentDate;

@FXML
private TextField txSalary;

@FXML
private TableColumn<UserDto, String> columnLogin;

@FXML
private TableView<UserDto> userTable;

@FXML
private TableColumn<UserDto, UUID> columnUserId;

@FXML
private TableColumn<UserDto, String> columnName;

@FXML
private TableColumn<UserDto, String> columnDepartmentName;

@FXML
private TableView<PaymentDto> paymentTable;

@FXML
private TableColumn<UserDto, Date> paymentColumnDate;

@FXML
private TableColumn<UserDto, String> paymentColumnSalary;

@FXML
private TableColumn<UserDto, String> paymentColumnAward;

@FXML
private Pane pane;

@Lazy
@Autowired
private StageManager stageManager;

@Autowired
private UserService userService;
```

```
  private final ObservableList<UserDto> userList =
FXCollections.observableArrayList();

  private final ObservableList<PaymentDto> paymentDtoObservableList =
FXCollections.observableArrayList();

  @Override
  public void initialize(URL location, ResourceBundle resources) {
    clearFields();
    userTable.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);

    setUserColumnProperties();
    setAwardColumnProperties();

    loadUserDetails();
  }

  private void setUserColumnProperties() {
    columnUserId.setCellValueFactory(new PropertyValueFactory<>("id"));
    columnLogin.setCellValueFactory(new PropertyValueFactory<>("login"));
    columnName.setCellValueFactory(new PropertyValueFactory<>("name"));
    columnDepartmentName.setCellValueFactory(new
PropertyValueFactory<>("department"));
  }

  private void setAwardColumnProperties() {
    paymentColumnDate.setCellValueFactory(new
PropertyValueFactory<>("receiveDate"));
    paymentColumnSalary.setCellValueFactory(new
PropertyValueFactory<>("salary"));
    paymentColumnAward.setCellValueFactory(new
PropertyValueFactory<>("award"));
  }

  private void loadUserDetails() {
    userList.clear();
    userList.addAll(userService.getAll());
    userTable.setItems(userList);
    updateAwardTable(Collections.emptyList());
  }

  private void updateAwardTable(List<PaymentDto> paymentDtos) {
    paymentDtoObservableList.clear();
    paymentDtoObservableList.addAll(paymentDtos);
    paymentTable.setItems(paymentDtoObservableList);
    log.info(String.valueOf(paymentDtos));
  }

  @FXML
  private void logout() {
    stageManager.switchScene(FxmlView.LOGIN);
  }

  @FXML
  void reset() {
    clearFields();
    resetAwardFields();
  }

  @FXML
  void resetAwardFields() {
    txSalary.setText("");
    txPaymentDate.getEditor().setText(null);
```

```
  }

  @FXML
  public void clickedOnUserTable() {
    UserDto selectedItem = userTable.getSelectionModel().getSelectedItem();
    if (!Objects.isNull(selectedItem)) {
      boolean isAdmin = getUserRole(selectedItem).equals(UserRole.ADMIN);
      rbAdmin.setSelected(isAdmin);
      userId.setText(String.valueOf(selectedItem.getId()));
      txName.setText(selectedItem.getName());
      txDepartmentName.setText(selectedItem.getDepartment().getName());
      txLogin.setText(selectedItem.getLogin());
      txPassword.setText(selectedItem.getPassword());

      updateAwardTable(selectedItem.getPayments());
      updateDiagram(selectedItem.getPayments());
    }
  }

  @FXML
  private void saveUser() {
    if(userId.getText() == null || Objects.equals(userId.getText(), "")) {
      UserDto user = new UserDto();
      log.info("USER CREATED LOCAL");
      user.setRole(getUserRole(null));
      user.setName(txName.getText());
      user.setDepartment(new DepartmentDto(null,
txDepartmentName.getText()));
      user.setLogin(txLogin.getText());
      user.setPassword(txPassword.getText());
      log.info("USER CREATED FULL");
      UserDto newUser = userService.save(user);
      log.info("USER CREATED DB");
      saveAlert(newUser);
    } else {
      UserDto user = userService.getById(UUID.fromString(userId.getText()));

      user.setRole(getUserRole(null));
      user.setName(txName.getText());
      user.setDepartment(new DepartmentDto(null,
txDepartmentName.getText()));
      user.setLogin(txLogin.getText());
      user.setPassword(txPassword.getText());

      UserDto updatedUser = userService.save(user);
      updateAlert(updatedUser);
    }

    cleanDiagram();
    clearFields();
    loadUserDetails();
  }

  @FXML
  private void deleteUser() {
    Alert alert = new Alert(AlertType.CONFIRMATION);
    alert.setTitle("Confirmation Dialog");
    alert.setHeaderText(null);
    alert.setContentText("Are you sure you want to delete selected?");
    ButtonType action =
alert.showAndWait().orElseThrow(RuntimeException::new);

              if (action == ButtonType.OK) {
```

```
        userService.delete(userTable.getSelectionModel().getSelectedItem().get
Id());
                }
    loadUserDetails();
  }

  @FXML
  public void saveAward() {
    UserDto selectedItem = userTable.getSelectionModel().getSelectedItem();
    if(!Objects.isNull(selectedItem)) {
      Date date = Date.from(txPaymentDate.getValue().atStartOfDay()
          .atZone(ZoneId.systemDefault())
          .toInstant());
      long salary = Long.parseLong(txSalary.getText());
      PaymentDto paymentDto = new PaymentDto(date, salary,
getAwardBySalary(salary));
      selectedItem.getPayments().add(paymentDto);
      userService.save(selectedItem);
    }
    cleanDiagram();
    resetAwardFields();
    loadUserDetails();
  }

  private Long getAwardBySalary(long salary) {
    Double award = salary * AWARD_PERCENT;
    return award.longValue();
  }

  @FXML
  public void deleteAward() {
    PaymentDto paymentDto =
paymentTable.getSelectionModel().getSelectedItem();
    UserDto user = userTable.getSelectionModel().getSelectedItem();
    user.getPayments().remove(paymentDto);

    userService.save(user);

    resetAwardFields();
    loadUserDetails();
  }

  private void clearFields() {
    rbAdmin.setSelected(false);
    userId.setText(null);
    txName.clear();
    txDepartmentName.clear();
    txLogin.clear();
    txPassword.clear();
    updateAwardTable(Collections.emptyList());
  }

  private void saveAlert(UserDto user) {
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("User saved successfully.");
    alert.setHeaderText(null);
    alert.setContentText("The user " + user.getName() + " " + user.getLogin()
+ " has been created.");
    alert.showAndWait();
  }

  private void updateAlert(UserDto user) {
```

```
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("User updated successfully.");
    alert.setHeaderText(null);
    alert.setContentText("The user " + user.getName() + " " + user.getLogin()
+ " has been updated.");
    alert.showAndWait();
  }

  private void showExcelAlert() {
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Excel");
    alert.setHeaderText(null);
    alert.setContentText("temp.xlsx has created!");
    alert.showAndWait();
  }

  private UserRole getUserRole(UserDto userDto) {
    if(Objects.isNull(userDto)) {
      return rbAdmin.isSelected() ? UserRole.ADMIN : UserRole.BASIC;
    } else {
      return userDto.getRole();
    }
  }

  public void toExcel() {
    UserDto selectedItem = userTable.getSelectionModel().getSelectedItem();
    if (Objects.isNull(selectedItem)) {
      throw new RuntimeException();
    }
    userService.toExcel(selectedItem);
    showExcelAlert();
  }

  private void updateDiagram(List<PaymentDto> paymentDtos) {
    cleanDiagram();
    CategoryAxis categoryAxis = new CategoryAxis();
    categoryAxis.setLabel("Месяц");

    NumberAxis numberAxis = new NumberAxis();
    numberAxis.setLabel("Сумма");

    BarChart chart = new BarChart(categoryAxis, numberAxis);
    chart.setTitle("Месячная премия");

    XYChart.Series series = new XYChart.Series();
    series.setName("Премия");
    paymentDtos.forEach( paymentDto ->
        series.getData().add(new
XYChart.Data<>(getMonth(paymentDto.getReceiveDate()),
paymentDto.getAward())));

    chart.getData().add(series);
    pane.getChildren().add(chart);
  }

  private String getMonth(Date date) {
    return new SimpleDateFormat("MMMM", new Locale("ru")).format(date);
  }

  private void cleanDiagram() {
    pane.getChildren().clear();
  }
}
```

```
package by.bsuir.psp.server.service.impl;

import by.bsuir.psp.model.dto.PaymentDto;
import by.bsuir.psp.model.dto.UserDto;
import by.bsuir.psp.model.dto.service.UserService;
import by.bsuir.psp.server.mapper.UserMapper;
import by.bsuir.psp.server.model.User;
import by.bsuir.psp.server.repo.UserRepository;
import lombok.AllArgsConstructor;
import lombok.SneakyThrows;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellStyle;
import org.apache.poi.ss.usermodel.FillPatternType;
import org.apache.poi.ss.usermodel.IndexedColors;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFFont;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.springframework.stereotype.Service;

import java.io.File;
import java.io.FileOutputStream;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.Optional;
import java.util.UUID;
import java.util.stream.Collectors;

/**
 * DESCRIPTION
 *
 * @author Vladislav_Karpeka
 * @version 1.0.0
 */
@Service
@AllArgsConstructor
public class UserServiceImpl implements UserService {

  private final UserRepository repository;

  private final UserMapper mapper;

  @Override
  public UserDto authentication(String login, String password) {
    Optional<User> user = repository.findByLoginAndPassword(login, password);
    return user.map(mapper::userToDto).orElse(null);
  }

  @Override
  public UserDto save(UserDto userDto) {
   try {
     User dtoToUser = mapper.dtoToUser(userDto);
     User user = repository.save(dtoToUser);
     return mapper.userToDto(user);
   }catch (Exception e) {
     e.printStackTrace();
     return null;
   }
  }

  @Override
```

```
  public UserDto getById(UUID id) {
    return
mapper.userToDto(repository.findById(id).orElseThrow(RuntimeException::new));
  }

  @Override
  public UserDto getByLogin(String login) {
    return mapper.userToDto(repository.findByLogin(login));
  }

  @Override
  public List<UserDto> getAll() {
    List<User> users = (List<User>) repository.findAll();
    return
users.stream().map(mapper::userToDto).collect(Collectors.toList());
  }

  @Override
  public void delete(UUID id) {
    repository.deleteById(id);
  }

  @Override
  public void deleteAll(List<UserDto> list) {

repository.deleteAll(list.stream().map(mapper::dtoToUser).collect(Collectors.
toList()));
  }

  @SneakyThrows
  @Override
  public void toExcel(UserDto userDto) {
    Workbook workbook = new XSSFWorkbook();

    Sheet sheet = workbook.createSheet(userDto.getName());
    sheet.setColumnWidth(0, 10000);
    sheet.setColumnWidth(1, 10000);
    sheet.setColumnWidth(2, 10000);

    Row header = sheet.createRow(0);

    CellStyle headerStyle = workbook.createCellStyle();
    headerStyle.setFillForegroundColor(IndexedColors.LIGHT_BLUE.getIndex());
    headerStyle.setFillPattern(FillPatternType.SOLID_FOREGROUND);

    XSSFFont font = ((XSSFWorkbook) workbook).createFont();
    font.setFontName("Times New Roman");
    font.setFontHeightInPoints((short) 14);
    font.setBold(true);
    headerStyle.setFont(font);

    Cell headerCell = header.createCell(0);
    headerCell.setCellValue("Дата получения ЗП");
    headerCell.setCellStyle(headerStyle);

    headerCell = header.createCell(1);
    headerCell.setCellValue("ЗП");
    headerCell.setCellStyle(headerStyle);

    headerCell = header.createCell(2);
    headerCell.setCellValue("Премия");
    headerCell.setCellStyle(headerStyle);
```

```
    CellStyle style = workbook.createCellStyle();
    style.setWrapText(true);

    Row row;
    Cell cell;
    for (int i = 0; i < userDto.getPayments().size(); i++) {
      row = sheet.createRow(i+1);
      PaymentDto paymentDto = userDto.getPayments().get(i);

      cell = row.createCell(0);
      cell.setCellValue(new
SimpleDateFormat("dd/MM/yyyy").format(paymentDto.getReceiveDate()));
      cell.setCellStyle(style);

      cell = row.createCell(1);
      cell.setCellValue(paymentDto.getSalary());
      cell.setCellStyle(style);

      cell = row.createCell(2);
      cell.setCellValue(paymentDto.getAward());
      cell.setCellStyle(style);
    }

    File currDir = new File(".");
    String path = currDir.getAbsolutePath();
    String fileLocation = path.substring(0, path.length() - 1) + "temp.xlsx";

    FileOutputStream outputStream = new FileOutputStream(fileLocation);
    workbook.write(outputStream);
    workbook.close();
  }
}
```

```
package by.bsuir.psp.client.controller;

import by.bsuir.psp.client.config.StageManager;
import by.bsuir.psp.client.view.FxmlView;
import by.bsuir.psp.model.dto.UserDto;
import by.bsuir.psp.model.dto.UserRole;
import by.bsuir.psp.model.dto.service.UserService;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Lazy;
import org.springframework.stereotype.Controller;

import java.net.URL;
import java.util.Objects;
import java.util.Optional;
import java.util.ResourceBundle;

/**
 * @author Ram Alapure
 * @since 05-04-2017
 */

@Controller
public class LoginController implements Initializable {
```

```
  @Autowired
  private UserController userController;

  @FXML
  private Button btnLogin;

  @FXML
  private PasswordField password;

  @FXML
  private TextField username;

  @FXML
  private Label lblLogin;

  @Autowired
  private UserService userService;

  @Lazy
  @Autowired
  private StageManager stageManager;

  @FXML
  private void login() {
    UserDto userDto = userService.authentication(getUsername(),
getPassword());
    if (!Objects.isNull(userDto)) {

userController.setCurrentUserBasic(userDto.getRole().equals(UserRole.BASIC));
      stageManager.switchScene(FxmlView.USER);
    } else {
      lblLogin.setText("Ошибка входа.");
    }
  }

  public String getPassword() {
    return password.getText();
  }

  public String getUsername() {
    return username.getText();
  }

  @Override
  public void initialize(URL location, ResourceBundle resources) {

  }
}
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Листинг основных элементов программы

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package by.bsuir.psp.client.config;

import javafx.stage.Stage;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Lazy;

import java.io.IOException;
import java.util.ResourceBundle;

@Configuration
public class AppJavaConfig {

    @Autowired
    ApplicationContext ctx;

    @Bean
    public ResourceBundle resourceBundle() {
        return ResourceBundle.getBundle("Bundle");
    }

    @Bean
    @Lazy //Stage only created after Spring context bootstap
    public StageManager stageManager(Stage stage) throws IOException {
        SpringFXMLLoader springFXMLLoader = (SpringFXMLLoader)
ctx.getBean("springFXMLLoader");
        return new StageManager(springFXMLLoader, stage);
    }

}

package by.bsuir.psp.client.config;

import by.bsuir.psp.model.dto.service.UserService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.remoting.rmi.RmiProxyFactoryBean;

/**
 * DESCRIPTION
 *
 * @author Vladislav_Karpeka
 * @version 1.0.0
 */
@Configuration
public class RmiConfig {

  @Bean
```

```
  @Primary
  RmiProxyFactoryBean service() {
    RmiProxyFactoryBean rmiProxyFactory = new RmiProxyFactoryBean();
    rmiProxyFactory.setServiceUrl("rmi://localhost:1099/UserService");
    rmiProxyFactory.setServiceInterface(UserService.class);
    return rmiProxyFactory;
  }

  @Bean
  UserService userService(ApplicationContext context) {
    return context.getBean(UserService.class);
  }
}
```

```
package by.bsuir.psp.client.config;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.stereotype.Component;

import java.io.IOException;
import java.util.ResourceBundle;

/**
 * Will load the FXML hierarchy as specified in the load method and register
 * Spring as the FXML Controller Factory. Allows Spring and Java FX to
coexist
 * once the Spring Application context has been bootstrapped.
 */
@Component
public class SpringFXMLLoader {
    private final ResourceBundle resourceBundle;
    private final ApplicationContext context;

    @Autowired
    public SpringFXMLLoader(ApplicationContext context, ResourceBundle
resourceBundle) {
        this.resourceBundle = resourceBundle;
        this.context = context;
    }

    public Parent load(String fxmlPath) throws IOException {
        FXMLLoader loader = new FXMLLoader();
        loader.setControllerFactory(context::getBean); //Spring now FXML
Controller Factory
        loader.setResources(resourceBundle);
        loader.setLocation(getClass().getResource(fxmlPath));
        return loader.load();
    }
}
```

```
package by.bsuir.psp.client.config;

import by.bsuir.psp.client.view.FxmlView;
import javafx.application.Platform;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import org.slf4j.Logger;

import java.util.Objects;
```

```
import static org.slf4j.LoggerFactory.getLogger;

/**
 * Manages switching Scenes on the Primary Stage
 */
public class StageManager {

    private static final Logger LOG = getLogger(StageManager.class);
    private final Stage primaryStage;
    private final SpringFXMLLoader springFXMLLoader;

    public StageManager(SpringFXMLLoader springFXMLLoader, Stage stage) {
        this.springFXMLLoader = springFXMLLoader;
        this.primaryStage = stage;
    }

    public void switchScene(final FxmlView view) {
        Parent viewRootNodeHierarchy =
loadViewNodeHierarchy(view.getFxmlFile());
        show(viewRootNodeHierarchy, view.getTitle());
    }

    private void show(final Parent rootnode, String title) {
        Scene scene = prepareScene(rootnode);
        //scene.getStylesheets().add("/styles/Styles.css");

        //primaryStage.initStyle(StageStyle.TRANSPARENT);
        primaryStage.setTitle(title);
        primaryStage.setScene(scene);
        primaryStage.sizeToScene();
        primaryStage.centerOnScreen();

        try {
            primaryStage.show();
        } catch (Exception exception) {
            logAndExit ("Unable to show scene for title" + title,
exception);
        }
    }

    private Scene prepareScene(Parent rootnode){
        Scene scene = primaryStage.getScene();

        if (scene == null) {
            scene = new Scene(rootnode);
        }
        scene.setRoot(rootnode);
        return scene;
    }

    /**
     * Loads the object hierarchy from a FXML document and returns to root
node
     * of that hierarchy.
     *
     * @return Parent root node of the FXML document hierarchy
     */
    private Parent loadViewNodeHierarchy(String fxmlFilePath) {
        Parent rootNode = null;
        try {
            rootNode = springFXMLLoader.load(fxmlFilePath);
```

```
            Objects.requireNonNull(rootNode, "A Root FXML node must not be
null");
        } catch (Exception exception) {
            logAndExit("Unable to load FXML view" + fxmlFilePath, exception);
        }
        return rootNode;
    }


    private void logAndExit(String errorMsg, Exception exception) {
        LOG.error(errorMsg, exception, exception.getCause());
        Platform.exit();
    }

}


package by.bsuir.psp.client.view;

import java.util.ResourceBundle;

public enum FxmlView {

    USER {
        @Override
               public String getTitle() {
            return getStringFromResourceBundle("user.title");
        }

        @Override
               public String getFxmlFile() {
            return "/fxml/User.fxml";
        }
    },
    LOGIN {
        @Override
               public String getTitle() {
            return getStringFromResourceBundle("login.title");
        }

        @Override
               public String getFxmlFile() {
            return "/fxml/Login.fxml";
        }
    };

    public abstract String getTitle();
    public abstract String getFxmlFile();

    String getStringFromResourceBundle(String key){
        return ResourceBundle.getBundle("Bundle").getString(key);
    }

}



package by.bsuir.psp.model.dto.service;

import by.bsuir.psp.model.dto.UserDto;

import java.util.List;
import java.util.UUID;
```

```
public interface UserService {

    UserDto authentication(String login, String password);

    UserDto save(UserDto userDto);

    UserDto getById(UUID id);

    UserDto getByLogin(String login);

    List<UserDto> getAll();

    void delete(UUID id);

    void deleteAll(List<UserDto> list);

    void toExcel(UserDto userDto);
}
```

```
package by.bsuir.psp.model.dto;

import java.io.Serializable;
import java.util.List;
import java.util.UUID;

/**
 * DESCRIPTION
 *
 * @author Vladislav_Karpeka
 * @version 1.0.0
 */
public class UserDto implements Serializable {

  private UUID id;

  private UserRole role;

  private String login;

  private String name;

  private String password;

  private DepartmentDto department;

  private List<PaymentDto> payments;

  public UserDto() {
  }

  public UserDto(UUID id, UserRole role, String login, String name, String
password, DepartmentDto department, List<PaymentDto> payments) {
    this.id = id;
    this.role = role;
    this.login = login;
    this.name = name;
    this.password = password;
    this.department = department;
    this.payments = payments;
  }
```

```
  public UUID getId() {
    return id;
  }

  public void setId(UUID id) {
    this.id = id;
  }

  public UserRole getRole() {
    return role;
  }

  public void setRole(UserRole role) {
    this.role = role;
  }

  public String getLogin() {
    return login;
  }

  public void setLogin(String login) {
    this.login = login;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getPassword() {
    return password;
  }

  public void setPassword(String password) {
    this.password = password;
  }

  public DepartmentDto getDepartment() {
    return department;
  }

  public void setDepartment(DepartmentDto department) {
    this.department = department;
  }

  public List<PaymentDto> getPayments() {
    return payments;
  }

  public void setPayments(List<PaymentDto> payments) {
    this.payments = payments;
  }
}
package by.bsuir.psp.model.dto;

import java.io.Serializable;
import java.util.Date;
import java.util.UUID;
```

```
/**
 * DESCRIPTION
 *
 * @author Vladislav_Karpeka
 * @version 1.0.0
 */
public class PaymentDto implements Serializable {

  private UUID id;

  private Date receiveDate;

  private Long salary;

  private Long award;

  public PaymentDto() {
  }

  public PaymentDto(UUID id, Date receiveDate, Long salary, Long award) {
    this.id = id;
    this.receiveDate = receiveDate;
    this.salary = salary;
    this.award = award;
  }

  public PaymentDto(Date receiveDate, Long salary, Long award) {
    this.receiveDate = receiveDate;
    this.salary = salary;
    this.award = award;
  }

  public UUID getId() {
    return id;
  }

  public void setId(UUID id) {
    this.id = id;
  }

  public Date getReceiveDate() {
    return receiveDate;
  }

  public void setReceiveDate(Date receiveDate) {
    this.receiveDate = receiveDate;
  }

  public Long getSalary() {
    return salary;
  }

  public void setSalary(Long salary) {
    this.salary = salary;
  }

  public Long getAward() {
    return award;
  }

  public void setAward(Long award) {
    this.award = award;
  }
```

```
}


package by.bsuir.psp.model.dto;

import java.io.Serializable;
import java.util.UUID;

/**
 * DESCRIPTION
 *
 * @author Vladislav_Karpeka
 * @version 1.0.0
 */
public class DepartmentDto implements Serializable {

  private UUID id;

  private String name;

  public DepartmentDto() {
  }

  public DepartmentDto(UUID id, String name) {
    this.id = id;
    this.name = name;
  }

  public UUID getId() {
    return id;
  }

  public void setId(UUID id) {
    this.id = id;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  @Override
  public String toString() {
    return name;
  }
}
```