Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Распределенные системы обработки информации»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент кафедры экономической информатики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П.Лыщик |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«АВТОМАТИЗИРОВАННАЯ СИСТЕМА РАСЧЕТА СТИПЕНДИЙ СТУДЕНТОВ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 074002  Козлова Александра Андреевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**РЕФЕРАТ**

**СОДЕРЖАНИЕ**

[Перечень условных обозначений, символов и терминов 6](#_Toc121975071)

[Введение 8](#_Toc121975072)

[1 Анализ и моделирование системы расчета стипендий в ВУЗе 9](#_Toc121975073)

[1.1 Описание системы расчета стипендий в ВУЗе 9](#_Toc121975074)

[1.2 Разработка функциональной модели системы расчета стипендий в ВУЗе. 16](#_Toc121975075)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 20](#_Toc121975076)

[1.4 Разработка информационной модели системы расчета стипендий в ВУЗе. 22](#_Toc121975077)

[1.5 Модели представления программного средства и их описание 24](#_Toc121975078)

[2 Проектирование и конструирование программного средства 28](#_Toc121975079)

[2.1 Постановка задачи 28](#_Toc121975080)

[2.2 Архитектурные решения 29](#_Toc121975081)

[2.3 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства 31](#_Toc121975082)

[2.4 Проектирование пользовательского интерфейса 34](#_Toc121975083)

[2.5 Обоснование выбора компонентов и технологий для реализации программного средства 35](#_Toc121975084)

[3 Тестирование и проверка работоспособности программного средства 39](#_Toc121975085)

[4 Инструкция по развертыванию и использованию программного средства 41](#_Toc121975086)

[4.1 Инструкция по развертыванию приложения 41](#_Toc121975087)

[4.2 Сквозной тестовый пример 42](#_Toc121975088)

[Заключение 60](#_Toc121975089)

[Список использованных источников 61](#_Toc121975090)

[Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» 63](#_Toc121975091)

[Приложение Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику 64](#_Toc121975092)

[Приложение В (обязательное) Листинг скрипта генерации базы данных 76](#_Toc121975093)

[Приложение Г (обязательное) Схемы алгоритмов программы 79](#_Toc121975094)

[Ведомость документов курсового проекта 81](#_Toc121975095)

# **Перечень условных обозначений, символов и ТЕРМИНОВ**

# **ВВЕДЕНИЕ**

В современном мире организации, хранящие данные о большом количестве людей, нуждаются в мощной поддержке со стороны автоматизированных систем. Большой объем информации делает затруднительным и длительным процесс обработки и дальнейшего использования данных.

Высшие учебные заведения хранят в своих базах личную информацию о каждом студенте, данные о его форме обучения, успеваемости, посещаемости, научной, общественной и спортивной деятельности. Четкая структура ВУЗа (разделение на профили обучения, факультеты и на специальности внутри факультетов) требует такой же четкой системы соотношения данных о специальностях и предметах с данными о студенте.

Одна из главных функций такой системы, помимо учета студентов, это расчет учебной стипендии для студентов бюджетной формы. Данный процесс зависит от корректности информации, хранимой в базе данных, а любые финансовые операции требуют повышенной точности.

Выбранная тема является актуальной, так как сфера образования всегда являлась и до сих пор является одной из важнейших в обществе. Базовые знания для дальнейшей жизни получаются именно в студенческое время. От качества функционирования информационного обеспечения зависит и качество процесса получения образования студентом.

*Целью курсового проекта* является уменьшение трудовых и временных затрат на учет студентов и расчет их стипендий за счет создания автоматизированной системы расчета студенческих стипендий.

Для корректного решения поставленной цели следует поставить и решить следующие задачи:

* исследование основных аспектов учета студентов в высшем учебном заведении, выполнение моделирования процесса расчета стипендий, определение требований к системе, организация работы с базой данных, разработка моделей представления системы;
* разработка архитектурных решений для приложения, разработка программы, которая позволит автоматически осуществлять расчет стипендий, разработка интерфейса окна, описание основных алгоритмов программы, обоснование выбора компонентов и технологий для разрабатываемого приложения;
* тестирование программного средства;
* описание работы приложения и инструкции по развертыванию.

Объектом исследования данной курсовой работы являются процесс расчета студенческой учебной стипендии.

Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников».

1. **АНАЛИЗ И МОДЕЛИРОВАНИЕ СИСТЕМЫ РАСЧЕТА СТИПЕНДИЙ В ВУЗЕ** 
   1. **Описание системы расчета стипендий в ВУЗе**

Предметной областью данного курсового проекта является система расчета стипендий в высшем учебном заведении (ВУЗе).

Все студенты ВУЗа делятся на две основные категории: обучающиеся за счет средств республиканского бюджета (студенты бюджетной формы) и обучающиеся за счет собственных средств (студенты платной формы). В соответствии с принятыми в стране указами [1], стипендии и надбавки к ним получают только студенты бюджетной формы образования при соблюдении следующих условий:

* отсутствие академических задолженностей (в случае пересдачи экзамена на сессии студент лишается стипендии на весь последующий семестр);
* удовлетворительная посещаемость (не более 10 часов пропусков в месяц, предшествующий месяцу выплаты стипендии, в против случае – лишение стипендии на месяц).

В конце каждого семестра по результатам экзаменационной сессии на основании сформированного среднего балла студенту бюджетной формы назначается учебная стипендия. Размер стипендии рассчитывается как базовая стипендия, умноженная на коэффициент, соответствующий среднему баллу студента. Все студенты первого курса бюджетной формы в первом полугодии получают базовую стипендию, которая, в свою очередь, рассчитывается как величина бюджета прожиточного минимума, умноженного на коэффициент 0,3 (для студентов, осваивающих содержание образовательных программ высшего образования I ступени) [2].

Величины коэффициентов при различных средних баллах так же зависят от профиля специальности, осваиваемой студентом. Также в Республике Беларусь студенты некоторых отдельных университетов получают рассчитанную по общим правилам стипендию, умноженную на определенный для ВУЗа коэффициент. Например, студенты БГУ и Академии управления при Президенте Республики Беларусь получают учебную стипендию, умноженную на 1,3.

Если учебные стипендии рассчитываются индивидуально для каждого студента на основании получаемой квалификации и успеваемости, то именные и президентские стипендии назначаются в определенном количестве в специальных постановлениях и заранее распределяются между университетами.

Так, согласно указу Президента Республики Беларусь «О социальной поддержке обучающихся» [2], учреждено 200 стипендий Президента Республики Беларусь студентам, курсантам, слушателям, осваивающим содержание образовательных программ высшего образования I и II ступеней. Такие стипендии студенты получают дважды в год вдобавок к учебной стипендии.

Аналогичное постановление определяет распределение именных стипендий между университетами Беларуси [3].

Получается, что для расчета размера стипендии отдельного студента бюджетной формы нужно знать следующую информацию:

* профиль специальности;
* информация о посещаемости;
* информация о наличии/отсутствии академических задолженностей;
* информация об особых стипендиях.

Система расчета стипендий студентов ВУЗов – это механизм определения размера стипендии каждого отдельного студента бюджетной формы на основании его достижений и успеваемости. Входными данными для нее является обозначенная выше информация о каждом студенте, а выходными – размер начисляемой стипендии для конкретного студента.

Целью данной системы является оптимизация процесса начисления стипендий и тем самым увеличение потока студентов.

В системе две роли: администратор и клиент. Цель администратора – отслеживать процесс работы системы, обновлять информацию о студентах и правилах начисления стипендии для корректной работы системы.

Администратор управляет:

* данными о клиентах: добавляет/удаляет клиента, обновляет/просматривает информацию о клиенте (CRUD-операции);
* данными, необходимыми для расчета стипендий: вносит размер базовой стипендии, размер коэффициентов для расчета стипендии по успеваемости, размер особых стипендий;
* данными о специальностях и предметах (CRUD-операции);
* данными о посещаемости и успеваемости отдельного студента.

Клиентом является студент высшего учебного заведения. У него возможностей меньше, чем у администратора:

* просмотр информации об успеваемости;
* просмотр информации о посещаемости;
* просмотр личной информации;
* изменение личной информации;
* просмотр размера стипендии;
* изменение пароля от своего аккаунта.

Данные для расчета стипендий хранятся в отдельной базе данных. Расчет стипендии происходит автоматически на основе входных данных о студентах и правилах начисления стипендии, а размеры особых стипендий регулируются администратором системы.

Когда студента зачисляют в университет, для него создается аккаунт в общей системе, логином для аккаунта является номер студенческого билета, пароль может быть установлен студентом самостоятельно (по умолчанию – также номер студенческого билета). Аккаунт студента содержит следующую личную информацию:

* номер студенческого билета;
* фамилия, имя, отчество;
* дата рождения;
* форма получения образования;
* специальность;
* адрес электронной почты;
* номер телефона.

Как уже упоминалось выше, студент может изменить личную информацию о себе, а также изменить пароль (для изменения пароля необходимо подтвердить то, что студент является владельцем аккаунта – необходимо ввести текущий пароль).

Для каждого студента, которому подлежит начислять стипендию, создается экземпляр сущность, содержащий следующие поля:

* номер студенческого билета;
* размер учебной стипендии;
* размер именной стипендии (опционально);
* размер социальной стипендии (опционально);
* размер президентской стипендии (опционально).

По умолчанию студент бюджетной формы получается студенческую стипендию, при особых заслугах – именную или президентскую, при тяжелом социальном положении – социальную.

* 1. **Разработка функциональной модели системы расчета стипендий в ВУЗе**

Для функционального моделирования системы расчета студенческих стипендий был выбран стандарт IDEF0.

IDEF0 – графическая нотация, используемая для моделирования функциональных систем. IDEF0-диаграмма отражает логические отношения между составляющими системы. Для этого используются следующие составляющие: блоки работ, потоки входа, потоки выхода, управление и механизм. Блок работы обычно имеет в названии глагол или отглагольное существительное и обозначает процесс – часть функциональной системы. Поток входа обозначает средства и информацию, необходимую для преобразования и создания потока выхода – конечного продукта для данного блока. Механизм представляет собой средство или человека, выполняющего блок работы, а управление – постановления, документы, стандарты и прочие регламентирующие процесс составляющие.

На рисунке 1.2 представлена контекстная диаграмма верхнего уровня, содержащая функциональную модель «Автоматизировать расчет стипендий»:

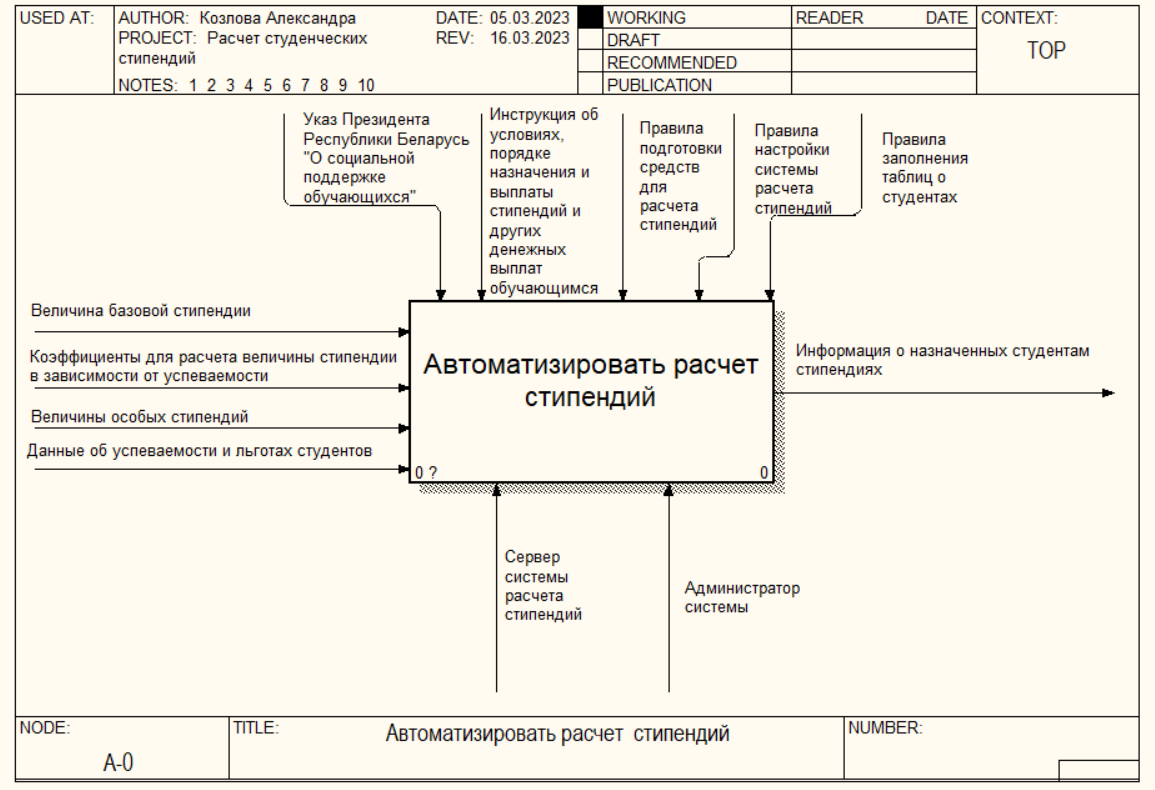


Рисунок 1.2 – Контекстная диаграмма верхнего уровня

Бизнес-процесс, представленный на диаграмме, имеет потоки входа – величину базовой стипендии, коэффициенты для расчета стипендий, величины особых стипендий и данные о студентах – которые в процессе преобразуются в поток выхода – информация о назначенных студентам стипендиях, предоставление чего и является целью автоматизированной системы расчета студенческих стипендий. Управление представлено правилами подготовки и настройки программного средства и различными постановлениями с правилами начисления стипендий студентам. Механизмом является сервер автоматизированной системы и администратор.

Далее на рисунке 1.3 представлена декомпозиция бизнес-процесса, состоящая из четырех блоков: «Подготовить программное средство», «Настроить систему расчета стипендий», «Заполнить таблицы данными о студентах» и «Вычислить размеры стипендий»:

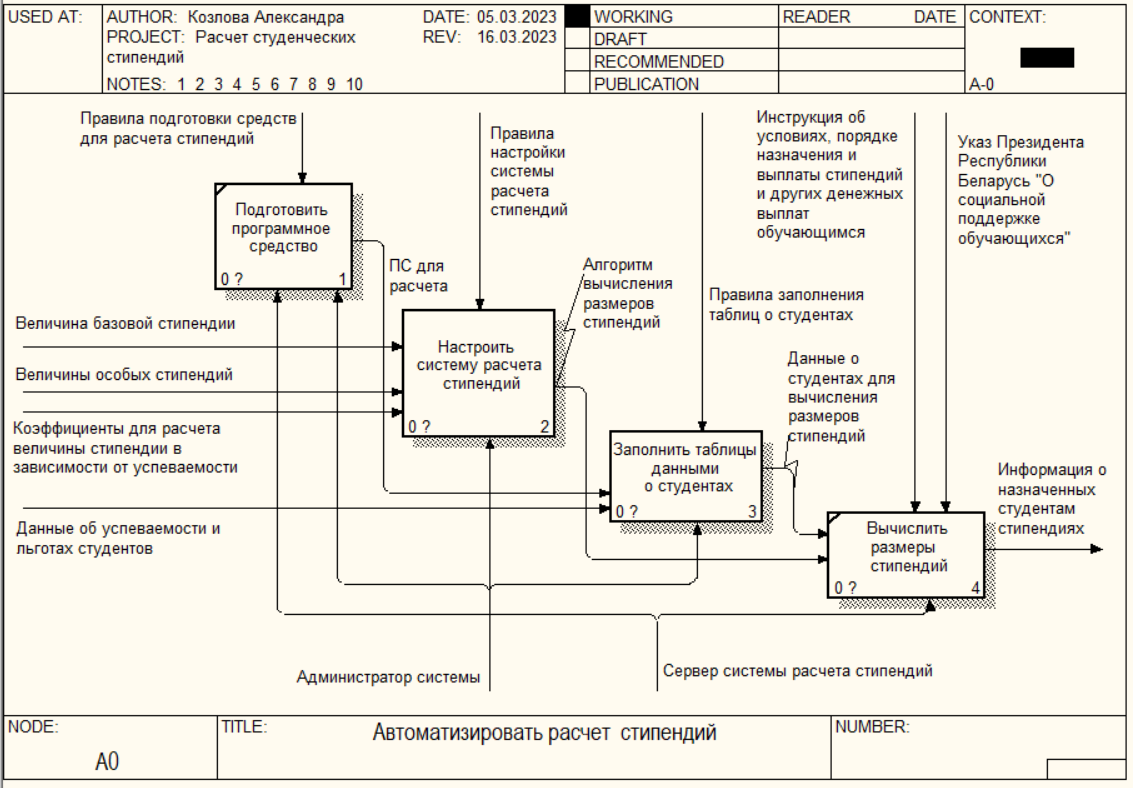


Рисунок 1.3 – Декомпозиция бизнес-процесса

При выполнении данных работ подготавливается программное средство и алгоритм для расчета стипендий, а также составляются таблицы с данными о студентах, на основе которых работает данный алгоритм.

Декомпозируем блок «Настроить систему расчета стипендий», чтобы подробнее рассмотреть процесс задачи алгоритма для подсчета стипендий (рисунок 1.4):

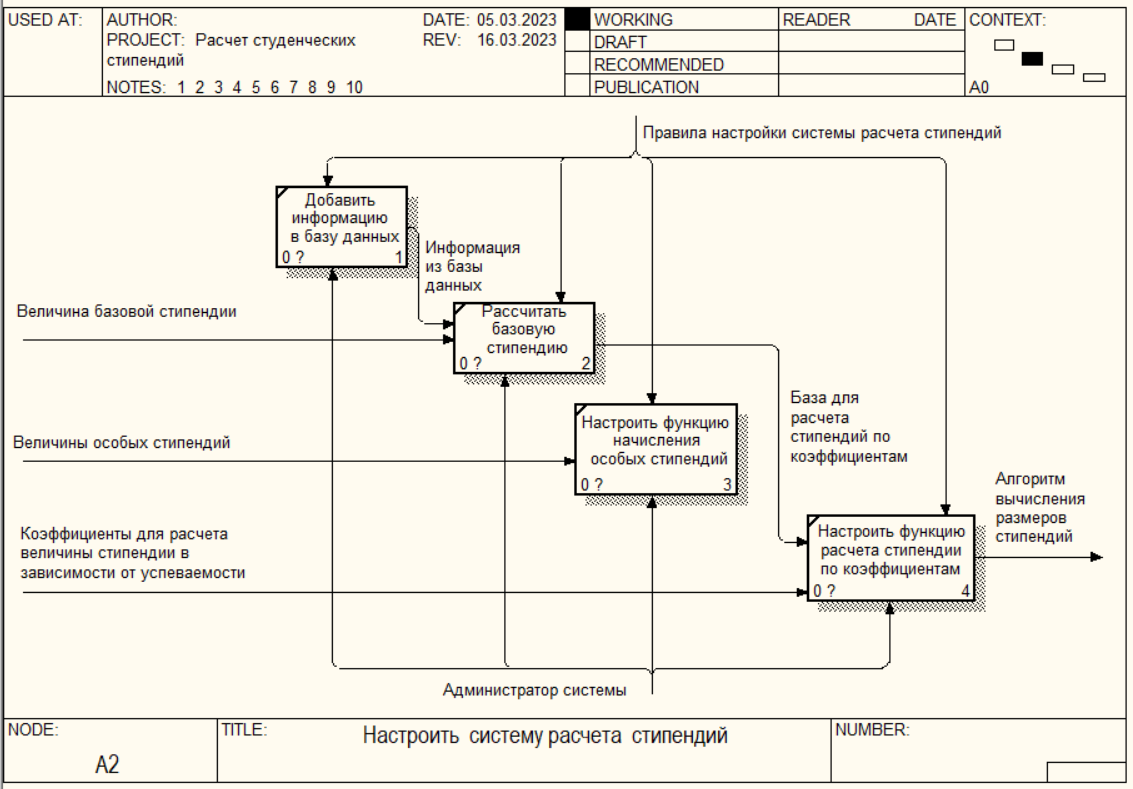


Рисунок 1.4 – Декомпозиция блока «Настроить систему расчета стипендий»

При декомпозиции этого блока мы видим, что создание алгоритма для вычисления размеров стипендий состоит из расчета базовой стипендии, задания функций умножения на коэффициенты и функции начисления особых стипендий (например, именных или стипендий Президента). Вся информация для расчетов стипендий заносится и сохраняется в базе данных.

Также декомпозируем блок «Заполнить таблицы данными о студентах», чтобы подробно рассмотреть, какие таблицы будут задействованы в системе расчета стипендий (рисунок 1.5):

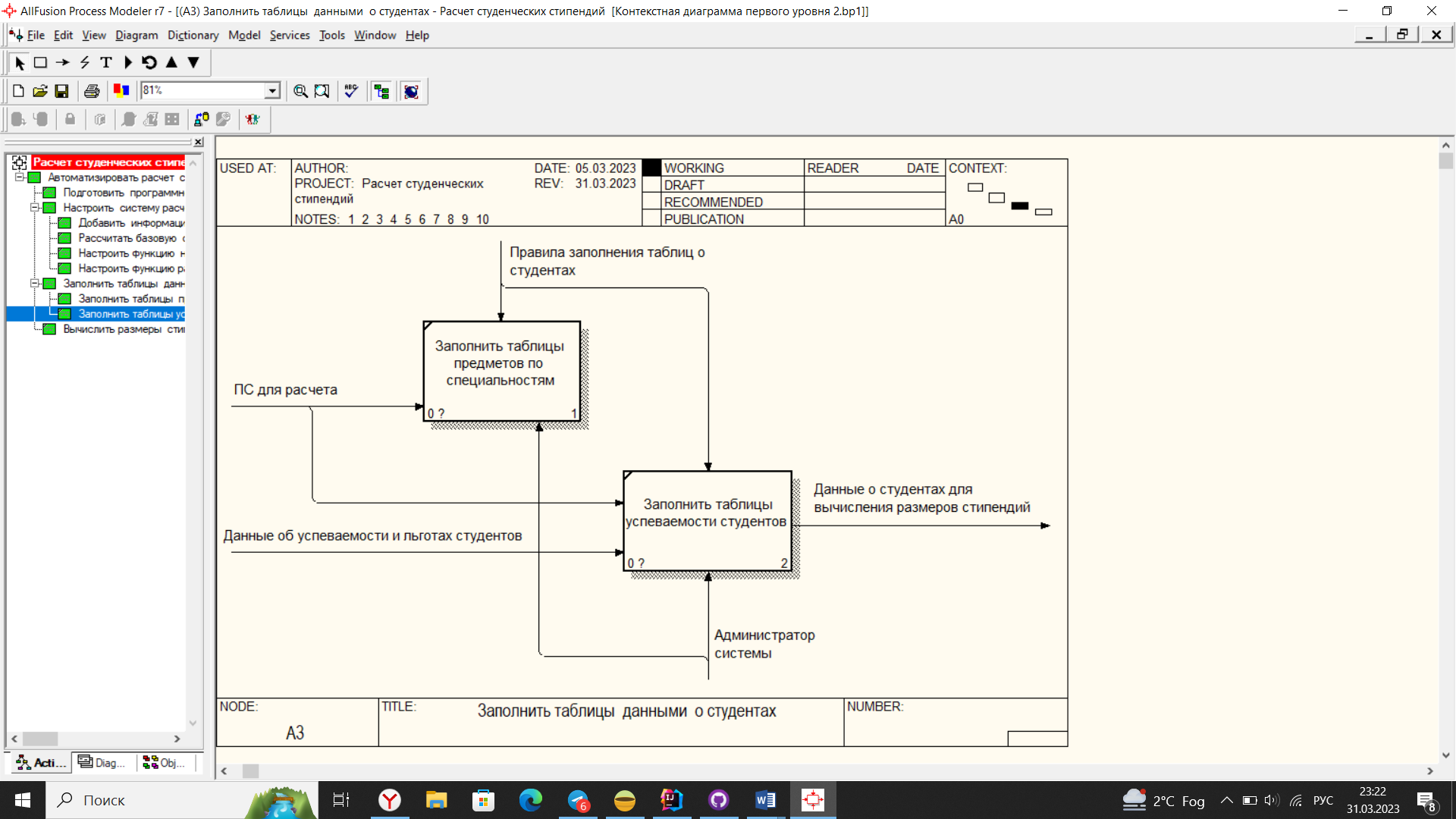


Рисунок 1.5 – Декомпозиция блока «Заполнить таблицы данными о студентах»

При декомпозиции данного блока мы видим, что на основе таблиц предметов по специальностям создаются и заполняются таблицы успеваемости студентов. Выходной поток этого блока является входным для блока «Вычислить размеры стипендий», который является завершающим в декомпозиции бизнес-процесса, то есть на основе данных таблиц на выходе из системы расчета стипендий получается информация о каждом студенте и размере его стипендии.

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

Для составления требований к разрабатываемому программному средству необходимо выделить несколько важных составляющих процесса анализа.

На этапе проектирования информационной системы формируют MVP – Minimum Viable Product (МЖП, минимальный жизнеспособный продукт). Это самая ранняя версия продукта, у которой есть минимальный набор функций, достаточный для презентации публике и проверке на первых потребителях. Однако такой продукт обязан демонстрировать достаточную ценность для пользователей.

Для описания функциональности MVP используют пользовательские истории.

Пользовательские истории (англ. User Story) — способ описания требований к разрабатываемой системе, сформулированных как одно или более предложений на повседневном или деловом языке пользователя. Каждая пользовательская история ограничена в размере и сложности [4].

User story помогают определить, какие функциональности необходимо включить в MVP продукта, а какие можно реализовать на следующих этапах, что помогает расставить приоритеты.

User story описывает роль пользователя в продукте, его потребность и результат, который он получит, если событие произойдет. Для быстроты составления используется следующий шаблон: «Я как ***(тип пользователя)***, хочу ***(действие или цель пользователя)***, чтобы ***(получить следующий результат или выгоду)***» [5].

Как уже упоминалось выше, в нашей системе две роли: администратор и клиент. Составим user stories для администратора:

1. Я как***администратор системы*** хочу ***выполнять CRUD-операции***, ***связанные со студентами***, чтобы ***поддерживать соответствие***данных в системе ***действительности*** и ***работать*** в последующем ***с актуальной информацией***.

2. Я как***администратор системы*** хочу ***вносить изменения в размер базовой стипендии и коэффициентов***, чтобы ***следовать актуальным правилам начисления стипендий***.

3. Я как***администратор системы*** хочу ***выполнять CRUD-операции***, ***связанные со специальностями***, чтобы ***данные в системе соответствовали перечню специальностей*** в университете.

4. Я как***администратор системы*** хочу ***выполнять CRUD-операции, связанные с предметами***, чтобы ***данные в системе соответствовали учебной программе и были актуальными***.

5. Я как***администратор системы*** хочу ***отслеживать посещаемость и успеваемость студентов***, чтобы ***правильно распределять им стипендии***.

Получается, что на основании данных требований администратор должен управлять данными о студентах, специальностях и предметах, данными о посещаемости и пропусках и размерами коэффициентов и базовой стипендии для правильного и актуального расчета стипендий. При редактировании информации о специальностях и предметах измененные данные должны обновляться и во всех связанных с ними полях других таблиц.

Теперь рассмотрим User story для клиента, т.е. студента.

1. Я как ***клиент системы*** хочу ***изменить пароль*** в случае необходимости, чтобы ***облегчить процедуру авторизации*** в системе.

2. Я как ***студент университета*** хочу ***иметь возможность просматривать расчет стипендии***, чтобы ***знать, почему у меня именно такой размер стипендии***.

3. Я как ***студент университета*** хочу ***иметь возможность редактировать личные данные***, чтобы ***администрация университета могла со мной связаться***.

Для наглядного отображения всех возможных операций была составлена диаграмма вариантов использования (use case diagram), которая позволяет наглядно изобразить основные возможности, доступные пользователю в разных ролях при работе с программой.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация функций, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий [6].

На рисунке 1.6 представлена диаграмма вариантов использования для разрабатываемой системы расчета стипендий студентов ВУЗов:

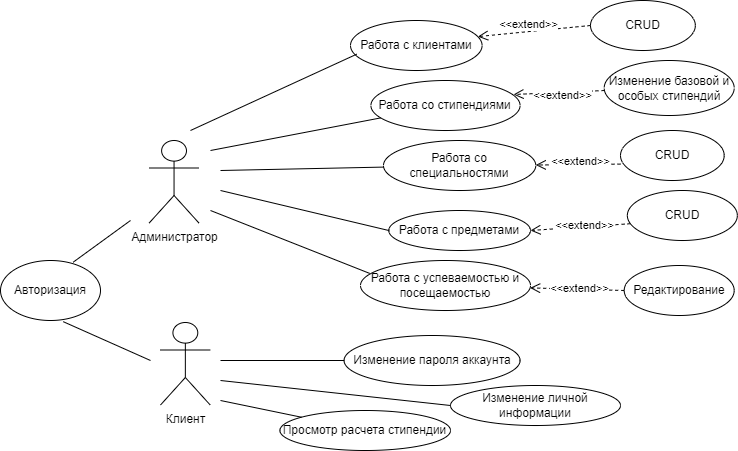


Рисунок 1.6 – Диаграмма вариантов использования

После успешной авторизации в зависимости от роли пользователь получается возможность совершать действия, доступные для его роли. Клиент может просматривать и редактировать личные данные, изменять пароль и просматривать расчет стипендии. Администратор работает с таблицами предметов, специальностей, студентов – добавляет, удаляет, просматривает и редактирует, а также имеет возможность изменять размеры стипендий.

Построенная диаграмма вариантов использования позволяет наглядно отразить применение разработанного программного продукта.

* 1. **Разработка информационной модели системы расчета стипендий в ВУЗе**

Важнейшая цель проектирования информационной модели – выработка непротиворечивой структурированной интерпретации реально существующей информации изучаемой предметной области и взаимодействия между ее структурными компонентами.

Понятие концептуальной модели данных связано с методологией семантического моделирования данных, т.е. с представлением данных в контексте их взаимосвязей с другими данными. Для построения информационной модели системы расчета стипендий в ВУЗе была использована нотация IDEF1X.

Методология IDEF1X – один из подходов к семантическому моделированию данных, основанный на концепции "сущность-связь" (Entity-Relationship). Это инструмент для анализа информационной структуры систем различной природы. Информационная модель, построенная с помощью IDEF1X-методологии, отображает логическую структуру информации об объектах системы

Таким образом, концептуальная модель, представленная в соответствии со стандартом IDEF1X, является логической схемой базы данных для проектируемой системы. Основными объектами концептуальной модели являются сущности и связи [7].

На рисунке 1.7 представлена информационная модель для курсового проекта. Для разработки было использовано средство *MySQL Workbench*.

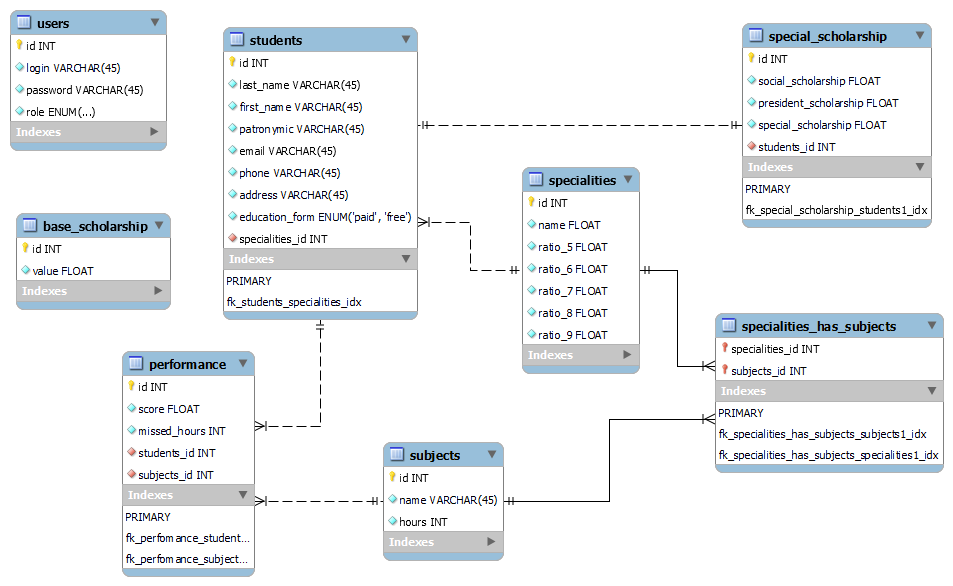


Рисунок 1.7 – Информационная модель системы расчета студенческих стипендий

В систему входят следующие сущности:

* students;
* specialities;
* subjects;
* users;
* performance;
* base\_scholarship;
* special\_scholarship;
* specialities\_has\_subjects.

Сущность **students** связана с сущностями **specialities** и **performance** связями «один-ко-многим» и **special\_scholarship** связью «один-к-одному» и хранит в себе данные о студентах (ФИО, id студента, адрес электронной почты, адрес проживания, форму получения образования, id специальности).

Сущность **specialities** связана с сущностью subjects связью «многие-ко-многим» (в модель добавлена промежуточная сущность **specialities\_has\_subjects**, связанная с сущностями **specialities** и **subjects** связями «один-ко-многим») и хранит в себе информацию о специальностях (id специальности, название, коэффициенты для различных средних баллов).

Сущность **subjects** связана с сущностью **performance** и хранит в себе информацию о предметах (id предмета, название, количество часов).

Сущность **performance** хранит в себе информацию о посещаемости студентом предметов (id посещаемости, средний балл, количество пропущенных часов, id студента, id предмета).

Сущность **special\_scholarship** связана с сущностью **students** связью «один-к-одному» и хранит в себе данные об особых стипендиях (id стипендии, размеры особой, президентской, именной и социальной стипендий).

Сущность **base\_scholarship** хранит в себе данные о базовой стипендии (id базовой стипендии и ее размер).

Сущность **users** хранит в себе данные о пользователях системы (id пользователя, логин, пароль, роль в системе).

Одно из требований к базе данных – быть приведенной к 3 нормальной форме (3NF). Требование третьей нормальной формы (3NF) заключается в том, чтобы в таблицах отсутствовала транзитивная зависимость.

Транзитивная зависимость – это когда неключевые столбцы зависят от значений других неключевых столбцов. Неключевые столбцы не должны пытаться играть роль ключа в таблице, такие столбцы не дают возможности получить данные из других столбцов, они дают возможность посмотреть на информацию, которая в них содержится, так как в этом их назначение [8].

* 1. **UML-модели представления программного средства и их описание**

***Диаграмма последовательности.*** Реализация отдельного варианта использования требует участия и взаимодействия определенных экземпляров актеров и классов. Наиболее подходящий инструмент для описания такого взаимодействия – это диаграммы последовательности и коммуникации, которые, по сути, отображают одну и ту же информацию.

Элементами диаграммы являются:

* экземпляры актеров и объекты, участвующие во взаимодействии;
* сообщения, передаваемые между экземплярами актеров и объектами.

Взаимодействие между экземплярами актеров и объектами моделируется посредством передачи сообщений. Сущность, отправляющую сообщение, называют клиентом, а принимающую – сервером. Таким образом, сообщения не только передают некоторую информацию, но и требуют или предполагают выполнения сервером определенных действий или передачу (возврат) клиенту необходимой информации. Прием сообщения обычно трактуется как возникновение события на сервере. Сообщения изображаются стрелкой с обязательным указанием направления (острие стрелки указывает на принимающую сторону) и спецификации [9].

Как уже говорилось выше, диаграмма последовательности строится для одного конкретного варианта использования. Ниже на рисунке 1.8 представлена диаграмма последовательности для варианта использования «Изменить пароль аккаунта» для пользователя «клиент».

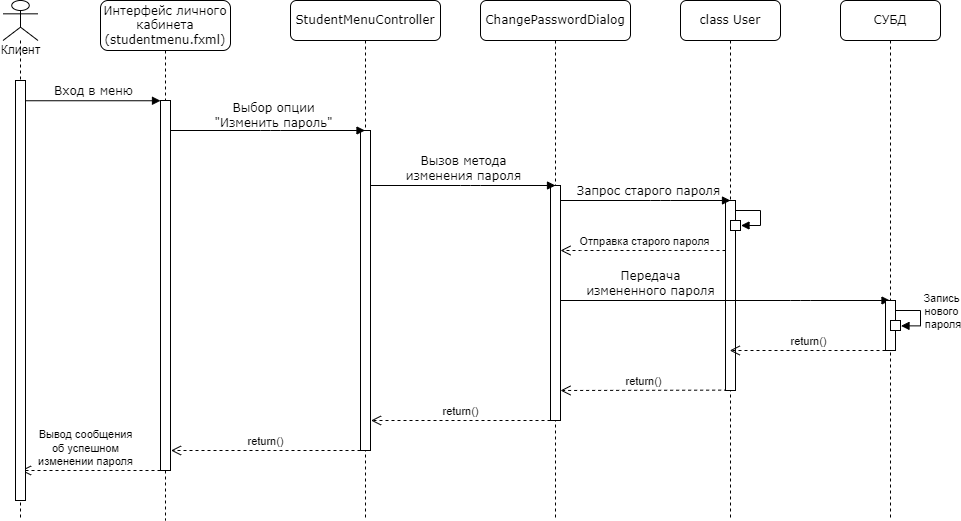


Рисунок 1.8 – Диаграмма последовательности для варианта использования «Изменить пароль аккаунта»

Сперва клиент выбирает опцию «Изменить пароль», чем вызывает класс StudentMenuController, в котором есть метод изменения пароля (появляется диалоговое окно). Для изменения необходимо подтвердить старый пароль, поэтому в методе происходит обращение к классу User, хранящему текущий пароль. После подтверждения старого пароля новый пароль записывается в базу данных, передается классу User, и клиент видит сообщение об успешном изменении пароля.

***Диаграмма состояний***. Данная диаграмма описывает возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий.

Хотя диаграммы состояний чаще всего используются для описания поведения отдельных экземпляров классов (объектов), но они также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершинами этого графа являются состояния, которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние [10].

На рисунке 1.9 представлена диаграмма состояний для варианта использования «Редактирование предмета» (одна из CRUD-операций).

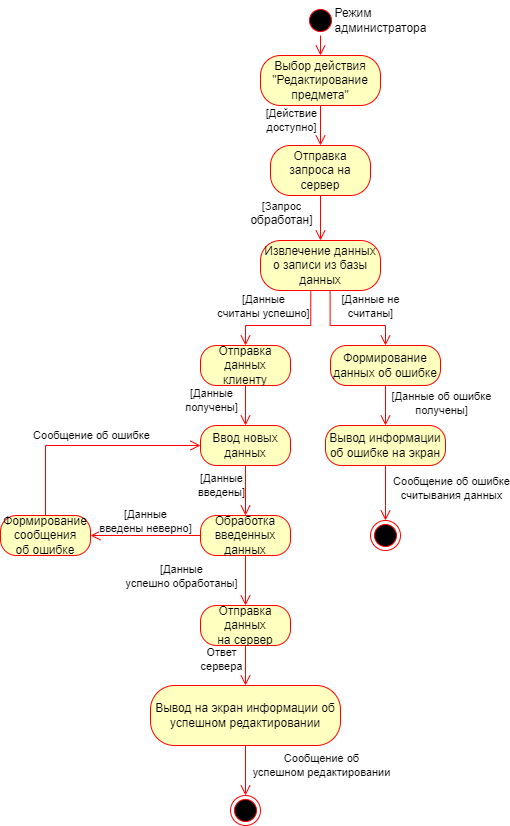


Рисунок 1.9 – Диаграмма состояний для варианта использования «Редактирование предмета»

Изначально система находится в режиме администратора, затем администратор выбирает пункт меню «редактировать предмет». Далее (при условии, что действие доступно) отправляется запрос о редактировании на сервер. После обработки запроса данные о записи извлекаются из базы данных. Если произошла ошибка при считывании данных, то формируются данные об ошибке и соответствующее сообщение выводится на экран. Если данные были успешно извлечены из базы данных, то они отправляются администратору. Далее он вводит новые данные. Если данные введены некорректно, система вернется в состояние «ввод новых данных», т.е. цикл будет продолжаться до ввода корректных данных. После успешного ввода данные обрабатываются и отправляются серверу, после чего на экран выводится сообщение об успешном редактировании данных. Система приходит в конечное состояние.

***Диаграмма развертывания***. Это тип диаграммы, который определяет физическое оборудование, на котором будет работать программная система. Диаграмма также определяет способ развертывания программного обеспечения на базовом оборудовании. Он отображает программные части системы на устройство, которое будет выполнять его.

Диаграмма развертывания отображает архитектуру программного обеспечения, созданную в проекте, в архитектуру физической системы, которая ее выполняет. В распределенных системах он моделирует распределение программного обеспечения по физическим узлам.

Программные системы проявляются с использованием различных артефактов, а затем они сопоставляются со средой выполнения, в которой будет выполняться программное обеспечение, например узлами. Многие узлы участвуют в диаграмме развертывания; следовательно, связь между ними представлена ​​с использованием путей связи [11].

На рисунке 1.10 представлена диаграмма развертывания разрабатываемого программного средства – системы расчета студенческих стипендий:

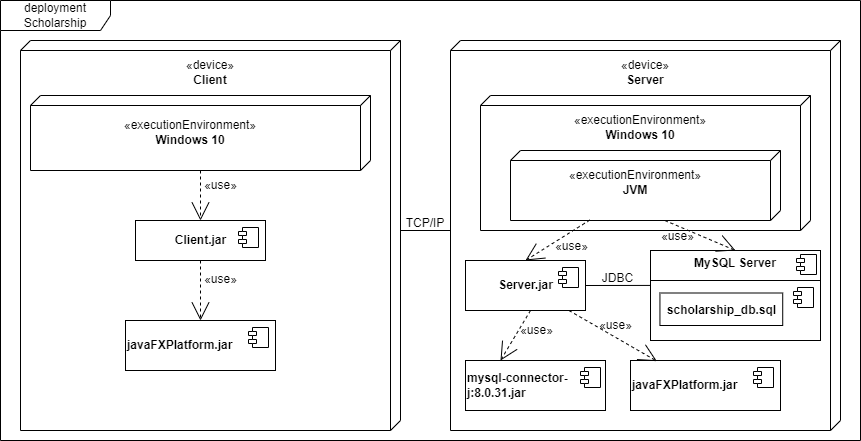


Рисунок 1.10 – Диаграмма развертывания системы расчета студенческих стипендий

На диаграмме представлено взаимодействие двух узлов: Client и Server, взаимодействующих на основе протокола TCP/IP. В каждом из них приведены использующиеся устройства.

***Узел Client***. Среда исполнения («executionEnvironment») с использованием Windows 10, которая не требует установления каких-либо дополнительных расширений. Тут используется такой компонент, как Client.jar, использующий, в свою очередь, javaFXPlatform.jar для работы с графическим интерфейсом.

***Узел Server***. Среда исполнения («executionEnvironment») с использованием Windows 10, включающая в себя «executionEnvironment» с использованием JVM (Java Virtual Machine, виртуальная машина Java). Использует компонент Server.jar, использующий, в свою очередь, mysql-connector-j:8.0.31.jar и javaFXPlatform.jar, а также MySQL Server, включающий в себя артефакт scholarship\_db.sql (база данных разрабатываемого программного средства). Для взаимодействия сервера приложения и СУБД используется стандартный API для языка Java – JDBC (Java Database Connectivity). Дополнительным компонентом, обеспечивающим подключение приложения к конкретной СУБД через JDBC API, является mysql-connector-j:8.0.31.

В данной главе были рассмотрены основные принципы работы системы расчета стипендий студентов высших учебных заведений, выполнено моделирование процесса «Автоматизировать расчет стипендий» (функциональная модель системы), определены требования к системе, разработана информационная модель базы данных, а также разработаны модели представления на основе стандарта UML 2.0.

1. **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**
   1. **Постановка задачи**

Задачей данного курсового проекта является разработка автоматизированной системы расчета стипендий студентов – многопоточного клиент-серверного приложения, взаимодействующего с базой данных.

Для реализации поставленных целей данного курсового проекта необходимо выполнить следующие задачи:

* изучить основные процессы предметной области – системы расчета стипендий;
* разработать приведенную к 3 нормальной форме базу данных (в качестве СУБД использовать MySQL);
* реализовать клиент-серверное приложение;
* связать сервер с базой данных;
* предусмотреть механизм авторизации;
* создать пользовательский интерфейс для удобной работы пользователя;
* сделать приложение простым и понятным в использовании.

Разработанное в ходе выполнения курсовой работы приложение будет доступно внутри университета.

Бизнес логика должна быть реализована только на серверной части приложения. Разработка системы предусмотрена для двух ролей: клиент (студент) и администратор. В зависимости от роли некоторые функции системы должны быть ограничены. У администратора должны быть права, которых нет у клиента, и наоборот.

Администраторы смогут регистрировать новых студентов, обновлять информацию о специальностях и предметах, а также заносить в базу данных новые размеры базовой и особых стипендий. Все внесенные изменения моментально вступают в работу. Данные могут удаляться, редактироваться, добавляться и обновляться администратором.

Пользователи могут зарегистрироваться в качестве студентов, просмотреть свою успеваемость, просмотреть расчет стипендии на основе успеваемости, изменить личную информацию и пароль от аккаунта.

Программное средство будет реализовано как клиент-серверное desktop-приложение на языке программирования Java с использованием СУБД MySQL. Просмотреть базу данных можно с помощью приложения MySQL Workbench. Для реализации графического интерфейса будет использоваться платформа JavaFX.

После решения вышеописанных задач клиент получит готовый программный продукт, а также документацию к нему, которая будет включать в себя: блок-схемы основных алгоритмов программы, UML-диаграммы, информационную модель, а также руководство пользователя.

* 1. **Обоснование выбора компонентов и технологий для реализации программного средства**

В процессе разработки программного продукта использовалась среда разработки IntelliJ IDEA Ultimate.

*IntelliJ IDEA* — ведущая IDE для разработки на Java и Kotlin. Она помогает работать продуктивнее за счет интеллектуальной помощи в написании кода, надежных рефакторингов, быстрой навигации по коду, широкого набора встроенных инструментов разработчика, поддержки веб- и корпоративной разработки и многих других полезных возможностей.

Преимущества IntelliJ IDEA:

* ***эргономичная среда*** (IntelliJ IDEA и готова к использованию сразу после установки, она обеспечивает быстрый доступ ко всем функциям и встроенным инструментам, необходимым разработчику, а также широкие возможности индивидуальной настройки);
* ***глубокий анализ кода*** (IntelliJ IDEA создавалась в первую очередь для разработки на Java и Kotlin, но она понимает многие другие языки программирования, в том числе Groovy, Scala, JavaScript, TypeScript и SQL, и предлагает интеллектуальную помощь в написании кода на каждом из них, она мгновенно обнаруживает ошибки, предлагает варианты автодополнения кода с учетом контекста, выполняет рефакторинг и т. д.);
* ***мгновенная навигация и поиск*** (в IntelliJ IDEA есть множество разнообразных функций для ускорения и упрощения навигации и поиска, которые помогают сосредоточиться на написании кода и работать быстрее);
* ***запуск, тестирование и отладка*** (в IntelliJ IDEA встроен эффективный набор инструментов для настройки параметров запуска и сборки приложения, отладки кода, а также применения и разработки тестов JUnit прямо в IDE);
* ***встроенные инструменты и интеграция*** (IntelliJ IDEA предлагает важные встроенные инструменты и возможности интеграции, благодаря которым можно работать в привычной среде, не переключаясь между разными приложениями);
* ***интеграция с системами контроля версий*** (IntelliJ IDEA по умолчанию поддерживает самые популярные системы контроля версий, такие как Git, Subversion, Mercurial и Perforce, проект из системы контроля версий можно клонировать прямо на начальном экране, проанализировать разницу между двумя версиями, управлять ветками, записывать и отправлять изменения, разрешать конфликты слияния, просматривать историю изменений и т. д.);
* ***поддержка фреймворков JVM*** (IntelliJ IDEA Ultimate обеспечивает первоклассную поддержку ведущих фреймворков и технологий для разработки современных приложений и микросервисов, в IDE встроена поддержка Spring и Spring Boot, Jakarta EE, JPA, Reactor и других фреймворков);
* ***веб-разработка*** (IntelliJ IDEA Ultimate включает в себя всю функциональность WebStorm — интегрированной среды разработки для JavaScript и связанных технологий, включая TypeScript, React, Vue, Angular, Node.js, HTML и файлов стилей);
* ***развертывание*** (чтобы не отставать от современных тенденции в разработке ПО, IntelliJ IDEA Ultimate предлагает интеграцию с самыми популярными системами управления контейнерами: Kubernetes и Docker);
* ***удаленная работа и совместная разработка*** (IntelliJ IDEA предлагает разработчикам все необходимое для удаленной работы, помогает продуктивно работать в команде из любой точки мира и писать код на ноутбуке любой мощности, ведь вся ресурсоемкая обработка выполняется на удаленном сервере) [16].

Для работы с базой данных была выбрана СУБД MySQL.

*MySQL* — это реляционная система управления базами данных (СУБД), которая распространяется как свободное программное обеспечение. Является одной из наиболее популярных, так как отличается гибкостью, легкостью, удобством в использовании.

Слово «реляционный» означает, что базы представлены в виде связанной информации и описываются как набор связей. MySQL работает с языком запросов SQL, который традиционно используется в базах данных.

Возможности MySQL:

* удобный доступ к базам данных;
* корректное распределенное хранение данных на сервере;
* быстрый поиск нужной информации в базе с помощью языка SQL;
* идентификация и обработка отдельных данных, их преобразование и отправка;
* создание, редактирование и удаление записей, которые есть в базе;
* отправка транзакций — «пакетов» из нескольких запросов к базе;
* множественный доступ к базе с разных устройств — например, из браузеров нескольких пользователей;
* выдача разных прав доступа различным пользователям;
* обеспечение безопасности данных: их защита, шифрование и контроль доступа;
* возможность контролировать версии базы данных: делать бэкапы, обновлять базу или откатывать назад;
* контроль состояния базы данных [17].

Для разработки пользовательского интерфейса была выбрана платформа JavaFX.

*JavaFX* – это библиотека для создания Java-приложений с графическим интерфейсом, насыщенных графикой, анимацией и интерактивными элементами [18].

С помощью JavaFX можно создавать программы для различных операционных систем: Windows, MacOS, Linux, Android, iOS и для самых различных устройств: десктопы, смартфоны, планшеты, встроенные устройства, ТВ. Приложение на JavaFX будет работать везде, где установлена исполняемая среда Java (JRE).

JavaFX 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 [19].

Для работы с JavaFX было выбрано приложение SceneBuilder.

В JavaFX *Scene Builder* — это инструмент, с помощью которого можно конструировать окна в виде графического интерфейса и после их сохранять, и эта программа на основании результата будет конструировать xml-файлы, которые будут подтягиваться в приложении [20].

* 1. **Архитектурные решения**

В данном подразделе будут описаны используемые в процессе создания программного средства паттерны проектирования и прочие архитектурные решения.

Паттерн проектирования – это часто встречающееся решение определённой проблемы при проектировании архитектуры программ.

В отличие от готовых функций или библиотек, паттерн нельзя просто взять и скопировать в программу. Паттерн представляет собой не какой-то конкретный код, а общую концепцию решения той или иной проблемы, которую нужно будет ещё подстроить под нужды вашей программы.

Паттерны часто путают с алгоритмами, ведь оба понятия описывают типовые решения каких-то известных проблем. Но если алгоритм — это чёткий набор действий, то паттерн — это высокоуровневое описание решения, реализация которого может отличаться в двух разных программах.

Один из применяемых паттернов – Builder. Это порождающий паттерн проектирования, который позволяет создавать сложные объекты пошагово. Строитель даёт возможность использовать один и тот же код строительства для получения разных представлений объекта [12].

В данный проект подключена основанная на аннотациях библиотека Lombok, позволяющая сократить шаблонный код. Используемая нами аннотация - @Builder. Встроенный в данную библиотеку Builder используется для сущностей, извлекаемых из базы данных. С помощью этой аннотации Lombok автоматически генерирует код, необходимый для создания экземпляров классов. Для этого используется метод builder(), неявно генерируемый Lombok`ом, который позже можно вызывать из класса, помеченного аннотацией @Builder, для создания экземпляра этого класса.

Еще один паттерн проектирования, используемый в данной работе, MVC.

MVC (Model-View-Controller) – это набор архитектурных идей и принципов для построения сложных систем с пользовательским интерфейсом. При разработке систем с пользовательским интерфейсом, следуя паттерну MVC, нужно разделять систему на три составные части: модель, представление и контроллер. Модель содержит всю бизнес-логику приложения. Представление (вид) отвечает за отображение данных пользователю. Все, что видит пользователь, генерируется видом. Третьим звеном данной цепи является контроллер. В нем хранится код, который отвечает за обработку действий пользователя (любое действие пользователя в системе обрабатывается в контроллере) [13].

В данном проекте будут соответствующие паттерну пакеты классов: пакет model с классами модели, которые также реализуют паттерн DAO (описан ниже), пакет controllers, в котором будут классы обработки полученных от пользователя данных, в качестве представления (View) в паттерне MVC будет использоваться GUI, написанный с помощью JavaFX.

Далее рассмотрим более подробно реализацию кода, связанного с работой с базой данных.

В данной работе за работу с базой данных отвечают DAO, JPA и Hibernate.

DAO – паттерн проектирования. Шаблон объекта доступа к данным (Data Access Object) – это структурный шаблон, который позволяет изолировать уровень приложения/бизнес-логики от уровня персистентности (обычно это реляционная база данных, но это может быть любой другой механизм персистентности) с помощью абстрактного API .

Функциональность этого API заключается в том, чтобы скрыть от приложения все сложности, связанные с выполнением операций CRUD в базовом механизме хранения. Это позволяет обоим слоям развиваться отдельно, ничего не зная друг о друге [14].

DAO обычно состоит из следующих компонентов:

* интерфейс DAO, который определяет набор методов для доступа к данным;
* конкретный класс DAO, который реализует интерфейс DAO и обеспечивает доступ к данным;
* модель данных, которая представляет данные, с которыми работает DAO.

JPA – Java Persistence API – стандартный интерфейс для работы с объектно-реляционной моделью данных (ORM) в Java-приложениях. Реализовать JPA можно с помощью следующих аннотаций:

* @Table (указывает таблицу в БД, в которой будут храниться объекты помеченного данной аннотацией класса);
* @Entity (указывает, что класс является сущностью, которую необходимо сохранить в БД);
* @Id (указывает, что поле является первичным ключом в БД);
* @Column (указывает имя столбца, в котором будут хранится значений помеченного данной аннотацией поля);
* @GeneratedValue (указывает, как генерировать значения для первичных ключей);
* @ManyToOne и @OneToMany (указывают как устанавливать связи между таблицами в БД);
* @Transient (указывает на то, что поле не должно сохраняться в базе данных) и т.д.

Описанные выше аннотации используются в данной курсовой работе.

Hibernate Framework — это фреймворк для языка Java, предназначенный для работы с базами данных. Он реализует объектно-реляционную модель — технологию, которая «соединяет» программные сущности и соответствующие записи в базе.

Объектно-реляционная модель, или ORM, позволяет создать программную «виртуальную» базу данных из объектов. Объекты описываются на языках программирования с применением принципов ООП. Hibernate — популярное воплощение этой модели.

Hibernate построен на спецификации JPA 2.1 — наборе правил, который описывает взаимодействие программных объектов с записями в базах данных. JPA поясняет, как управлять сохранением данных из кода на Java в базу. Но сама по себе спецификация — только теоретические правила, а в «чистой» Java ее реализации нет [15].

Подводя итог, можно сказать, что для работы с базой данных в данном проекте будут использоваться все три технологии. DAO используется для инкапсуляции доступа к данным, JPA – для маппинга Java-объектов на базу данных и управления жизненным циклом объектов, а Hibernate – как реализация JPA для работы с базой данных.

* 1. **Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства**

Основная функция разрабатываемого программного средства – расчет студенческой стипендии на основании посещаемости и успеваемости. Подсчет стипендии – алгоритм, реализующий ключевую бизнес-логику в данном программном средстве.

*Первым условием*, которое должно быть выполнено для начисления стипендии, является обучение на бесплатной форме. Если студент учится на платной, то алгоритм вернет сумму особых стипендий. В противном случае происходит расчет учебной стипендии.

*Второе условие* – количество часов пропусков за месяц, предшествующий месяцу расчета стипендии, должно быть меньше 10. В противном случае стипендия равна 0 (студент лишается стипендии на месяц).

Если данные условия соблюдены, то происходит расчет среднего балла по формуле среднего арифметического:

(2.1)

,

где – средний балл;

– сумма всех отметок по всем предметам;

– количество всех отметок по всем предметам.

* 1. **Проектирование пользовательского интерфейса**

**2.6 Методы и средства, используемые для обеспечения безопасности данных**

Один из важнейших моментов разработки программного средства – обеспечение безопасности данных.

***Хеширование паролей***. Пароль – условное слово или произвольный набор знаков, состоящий из букв, цифр и других символов, и предназначенный для подтверждения личности или полномочий. Это способ сохранения конфиденциальности и ограничения доступа к данным.

Авторизоваться в приложении можно только зная пароль. У каждого пользователя он свой. Все пароли хранятся в базе данных в таблице users. Для обеспечения безопасности и сохранности паролей используется хеширование. Именно результаты хеширования сохраняются в базе данных. При вводе пароля пользователем он хешируется и полученный результат сравнивается с хранящимся в базе данных значением.

*Криптографическая хеш-функция*, чаще называемая просто хешем, – это математический алгоритм, преобразовывающий произвольный массив данных в состоящую из букв и цифр строку фиксированной длины. Длина полученной строки будет оставаться неизменной, вне зависимости от объема вводных данных. Наиболее распространенная область применения хеширования – хранение паролей. Существует масса алгоритмов хеширования, отличающихся криптостойкостью, сложностью, разрядностью и другими свойствами [21].

*Secure Hash Algorithm (SHA)* – это семейство криптографических хеш-функций, которые используются большинством криптовалют. Они были разработаны Национальным институтом стандартов и технологий. Последний был выпущен в 2000 году. Считаются одними из самых безопасных алгоритмов хеширования.

В данном курсовом проекте будет использоваться хеш-функция SHA-512.

*SHA-512* – алгоритм дайджеста сообщений. Он используется для вычисления hash-значения в криптографии. Хеш-функция берет блок данных и возвращает битовую строку фиксированного размера (хешированные данные). Информацию, используемую хеш-функциями, называют «сообщением», а вычисленное значение хеш-функции – «дайджестом сообщения». SHA-512 выдает 512-битный (64 байта) хеш. Обычно это шестнадцатеричное число из 128 цифр [22].

Для реализации процесса хеширования в проекте будет использоваться библиотека *java.security* – часть стандартной библиотеки Java, которая предоставляет различные классы и интерфейсы, связанные с безопасностью, шифрованием, аутентификацией и управлением ключами.

Далее представлен листинг класса HashPassword, в котором прописан метод хеширования getHash(). В данном методе строка пароля преобразуется в массив байтов с помощью метода getBytes() и передается методу digest() объекта MessageDigest (часть библиотеки java.security), который вычисляет хэш-значение. Результатом является массив байтов, представляющий хэш-значение для заданного пароля.

public class HashPassword {

public static byte[] getHash(String password) throws NoSuchAlgorithmException, UnsupportedEncodingException

{

var digest = MessageDigest.getInstance("SHA-512");

digest.reset();

return digest.digest(password.getBytes(StandardCharsets.UTF\_8));

}

}

1. **ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**
2. **РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА**
   1. **Руководство по установке (развертыванию) программного средства**

Для того чтобы запустить приложение на любом компьютере, следует выполнить цепочку действий.

* 1. **Руководство пользователя**

# **ЗАКЛЮЧЕНИЕ**

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

# **Приложение А (обязательное)** **Отчет о проверке на заимствования в системе «Антиплагиат»**

Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

# **Приложение Б (обязательное)** **Листинг кода алгоритмов, реализующих основную бизнес-логику**

# **Приложение В (обязательное)** **Листинг скрипта генерации базы данных**

-- Дамп структуры для таблицы scholarship\_db.specialities  
CREATE TABLE IF NOT EXISTS `specialities` (  
 `id` *int* unsigned NOT NULL,  
 `name` *varchar*(255) DEFAULT NULL,  
 `ratio\_5` *float* unsigned DEFAULT NULL,  
 `ratio\_6` *float* unsigned DEFAULT NULL,  
 `ratio\_7` *float* unsigned DEFAULT NULL,  
 `ratio\_8` *float* unsigned DEFAULT NULL,  
 `ratio\_9` *float* unsigned DEFAULT NULL,  
 PRIMARY KEY (`id`),  
 UNIQUE KEY `id\_UNIQUE` (`id`)  
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci;  
  
-- Дамп данных для таблицы scholarship\_db.specialities  
INSERT INTO `specialities` (`id`, `name`, `ratio\_5`, `ratio\_6`, `ratio\_7`, `ratio\_8`, `ratio\_9`) VALUES  
(1, 'Электронный маркетинг', 0, 1, 1.1, 1.3, 1.5),  
(2, 'Программное обеспечение информационных технологий', 1, 1.2, 1.2, 1.4, 1.6),  
(3, 'Экономика электронного бизнеса', 0, 1, 1.1, 1.3, 1.5),  
(4, 'Информационные системы и технологии (в экономике)', 1, 1.2, 1.2, 1.4, 1.6);  
  
-- Дамп структуры для таблицы scholarship\_db.subjects  
CREATE TABLE IF NOT EXISTS `subjects` (  
 `id` *int* unsigned NOT NULL,  
 `name` *varchar*(45) DEFAULT NULL,  
 `hours` *int* unsigned DEFAULT NULL,  
 PRIMARY KEY (`id`),  
 UNIQUE KEY `id\_UNIQUE` (`id`)  
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci;  
  
-- Дамп данных для таблицы scholarship\_db.subjects  
INSERT INTO `subjects` (`id`, `name`, `hours`) VALUES  
(1, 'Основы алгоритмизации и программирования', 130),  
(2, 'Маркетинговые исследования', 140),  
(3, 'Программирование сетевых приложений', 100),  
(4, 'Экономическая теория', 90),  
(5, 'Эконометрика', 75);  
  
-- Дамп структуры для таблицы scholarship\_db.base\_scholarship  
CREATE TABLE IF NOT EXISTS `base\_scholarship` (  
 `id` *int* unsigned NOT NULL AUTO\_INCREMENT,  
 `value` *float* unsigned DEFAULT NULL,  
 PRIMARY KEY (`id`),  
 UNIQUE KEY `id\_UNIQUE` (`id`)  
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci;  
  
-- Дамп данных для таблицы scholarship\_db.base\_scholarship  
INSERT INTO `base\_scholarship` (`id`, `value`) VALUES  
(1, 112.58);  
  
-- Дамп структуры для таблицы scholarship\_db.specialities\_has\_subjects  
CREATE TABLE IF NOT EXISTS `specialities\_has\_subjects` (  
 `specialities\_id` *int* unsigned NOT NULL,  
 `subjects\_id` *int* unsigned NOT NULL,  
 PRIMARY KEY (`specialities\_id`,`subjects\_id`),  
 KEY `fk\_specialities\_has\_subjects\_subjects1\_idx` (`subjects\_id`),  
 KEY `fk\_specialities\_has\_subjects\_specialities1\_idx` (`specialities\_id`),  
 CONSTRAINT `fk\_specialities\_has\_subjects\_specialities1` FOREIGN KEY (`specialities\_id`) REFERENCES `specialities` (`id`) ON DELETE CASCADE,  
 CONSTRAINT `fk\_specialities\_has\_subjects\_subjects1` FOREIGN KEY (`subjects\_id`) REFERENCES `subjects` (`id`) ON DELETE CASCADE  
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci;  
  
-- Дамп данных для таблицы scholarship\_db.specialities\_has\_subjects  
INSERT INTO `specialities\_has\_subjects` (`specialities\_id`, `subjects\_id`) VALUES  
(1, 1), (1, 2), (1, 3), (2, 1), (2, 4), (3, 1), (3, 3), (3, 4), (3, 5), (4, 1), (4, 5);  
  
-- Дамп структуры для таблицы scholarship\_db.students  
CREATE TABLE IF NOT EXISTS `students` (  
 `id` *int* unsigned NOT NULL,  
 `last\_name` *varchar*(45) DEFAULT NULL,  
 `first\_name` *varchar*(45) DEFAULT NULL,  
 `patronymic` *varchar*(45) DEFAULT NULL,  
 `email` *varchar*(45) DEFAULT NULL,  
 `phone` *varchar*(45) DEFAULT NULL,  
 `address` *varchar*(45) DEFAULT NULL,  
 `education\_form` enum('paid','free') DEFAULT 'paid',  
 `specialities\_id` *int* unsigned NOT NULL,  
 PRIMARY KEY (`id`),  
 UNIQUE KEY `id\_UNIQUE` (`id`),  
 KEY `fk\_students\_specialities1\_idx` (`specialities\_id`),  
 CONSTRAINT `fk\_students\_specialities1` FOREIGN KEY (`specialities\_id`) REFERENCES `specialities` (`id`)  
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci;  
  
-- Дамп данных для таблицы scholarship\_db.students  
INSERT INTO `students` (`id`, `last\_name`, `first\_name`, `patronymic`, `email`, `phone`, `address`, `education\_form`, `specialities\_id` ) VALUES  
(1,'Козлова', 'Александра', 'Андреевна', 'kozlova.sasha2003@mail.ru', '+375293615057', 'ул. Семеняко, 4-12', 'free', 1),  
(2,'Бельская', 'Валерия', 'Алексеевна', 'valeryiabelskaya@gmail.com', '+375295963195', 'ул. Якуба Коласа, 28', 'free', 1),  
(3,'Марчик', 'Божена', 'Валерьевна', 'bozhenam@gmail.com', '+3752447124548', 'ул. Фанипольская, 1', 'free', 2),  
(4,'Зуёнок', 'Арина', 'Сергеевна', 'arinazuenok@mail.ru', '+375445103126', 'ул. Голубева, 36', 'paid', 3),  
(5,'Тимчук', 'Юлия', 'Сергеевна', 'jullytimchuk@mail.ru', '+375296102210', 'ул. Каменногорская, 12', 'free', 4);  
  
-- Дамп структуры для таблицы scholarship\_db.special\_scholarship  
CREATE TABLE IF NOT EXISTS `special\_scholarship` (  
 `id` *int* unsigned NOT NULL,  
 `social\_scholarship` *float* unsigned DEFAULT NULL,  
 `president\_scholarship` *float* unsigned DEFAULT NULL,  
 `special\_scholarship` *float* unsigned DEFAULT NULL,  
 PRIMARY KEY (`id`),  
 UNIQUE KEY `students\_id\_UNIQUE` (`id`),  
 KEY `fk\_special\_scholarship\_students1\_idx` (`id`),  
 CONSTRAINT `fk\_special\_scholarship\_students1` FOREIGN KEY (`id`) REFERENCES `students` (`id`) ON DELETE CASCADE  
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci;  
  
-- Дамп данных для таблицы scholarship\_db.special\_scholarship  
INSERT INTO `special\_scholarship` (`id`, `social\_scholarship`, `president\_scholarship`, `special\_scholarship`) VALUES  
(1, 90.06, 1290.6, 228.91);  
  
-- Дамп структуры для таблицы scholarship\_db.performance  
CREATE TABLE IF NOT EXISTS `performance` (  
 `id` *int* NOT NULL AUTO\_INCREMENT,  
 `total\_score` *float* unsigned DEFAULT NULL,  
 `missed\_hours` *int* unsigned DEFAULT NULL,  
 `students\_id` *int* unsigned NOT NULL,  
 `subjects\_id` *int* unsigned NOT NULL,  
 PRIMARY KEY (`id`),  
 UNIQUE KEY `id\_UNIQUE` (`id`),  
 KEY `fk\_performance\_students1\_idx` (`students\_id`),  
 KEY `fk\_performance\_subjects1\_idx` (`subjects\_id`),  
 CONSTRAINT `fk\_performance\_students1` FOREIGN KEY (`students\_id`) REFERENCES `students` (`id`) ON DELETE CASCADE,  
 CONSTRAINT `fk\_performance\_subjects1` FOREIGN KEY (`subjects\_id`) REFERENCES `subjects` (`id`) ON DELETE CASCADE  
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci;  
  
-- Дамп данных для таблицы scholarship\_db.performance  
INSERT INTO `performance` (`id`, `total\_score`, `missed\_hours`, `students\_id`, `subjects\_id`) VALUES  
(1, 9, 0, 1, 1), (2, 10, 0, 1, 2), (3, 9.2, 2, 1, 3),  
(4, 8, 0, 2, 1), (5, 9, 0, 2, 2), (6, 8.2, 0, 2, 3),  
(7, 8.4, 0, 3, 1), (8, 8.2, 0, 3, 4),  
(9, 9, 0, 4, 1), (10, 8, 0, 4, 3), (11, 8.75, 0, 4, 4), (12, 8.5, 0, 4, 5),  
(13, 9.1, 0, 5, 1), (14, 9.8, 0, 5, 5);  
  
-- Дамп структуры для таблицы scholarship\_db.users  
CREATE TABLE IF NOT EXISTS `users` (  
 `id` *int* unsigned NOT NULL AUTO\_INCREMENT,  
 `login` *varchar*(45) NOT NULL,  
 `password` *varchar*(45) NOT NULL,  
 `role` enum('Administrator','project.model.Student') NOT NULL DEFAULT 'project.model.Student',  
 PRIMARY KEY (`id`),  
 UNIQUE KEY `id\_UNIQUE` (`id`),  
 UNIQUE KEY `login\_UNIQUE` (`login`)  
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci;  
  
-- Дамп данных для таблицы scholarship\_db.users  
INSERT INTO `users` (`id`, `login`, `password`, `role`) VALUES  
(1, 'stud', 'stud', 'project.model.Student'),  
(2, 'admin', 'admin', 'Administrator');

# **Приложение Г (обязательное)** **Схема алгоритма, реализующего бизнес логику программы**

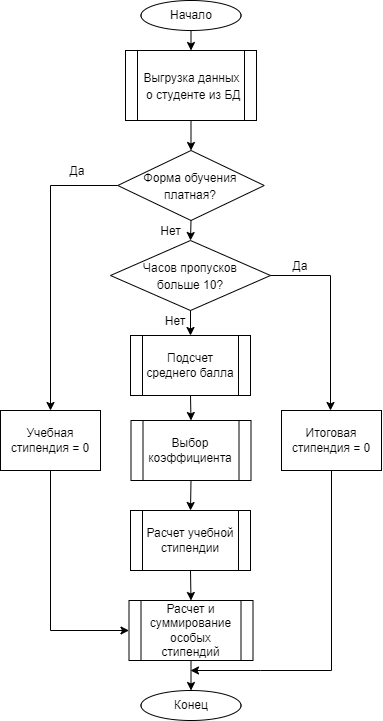


Рисунок Г.1 – Блок-схема алгоритма расчета стипендии

# **ВЕДОМОСТЬ ДОКУМЕНТОВ КУРСОВОГО ПРОЕКТА**