ТИТУЛЬНЫЙ

РЕФЕРАТ

БГУИР ДП 1-40 05 01-10 052 ПЗ

**Шинода М.Б.** Проектирование и разработка программного средства оптимизации процесса визовой поддержки иностранных студентов в отделе клиентского сопровождения университета: пояснительная записка к дипломному проекту / М.Б. Шинода – Минск : БГУИР, 2022. – 65 с.

Пояснительная записка 65 с., 18 рис., 5 табл., 20 источник, 3 приложения

ПРОГРАММНОЕ СРЕДСТВО ОПТИМИЗАЦИИ ПРОЦЕССА ВИЗОВОЙ ПОДДЕРЖКИ ИНОСТРАННЫХ СТУДЕНТОВ В ОТДЕЛЕ КЛИЕНТСКОГО СОПРОВОЖДЕНИЯ УНИВЕРСИТЕТА

*Цель проектирования*: разработка программного средства оптимизации процесса визовой поддержки иностранных студентов в отделе клиентского сопровождения университета.

*Методология проведения работы*: для решения поставленных задач были использованы принципы системного подхода, моделирование предметной области с использованием графических нотаций и стандартов, теория баз данных, принципы и подходы к проектированию оптимальной архитектуры программного обеспечения, современные практики написания эффективного программного кода, методики тестирования функциональности программного средства.

*Результаты работы*: выполнен анализ предметной области и существующих аналогов разрабатываемого программного средства; построена функциональная модель предметной области с использованием графических нотаций *IDEF0*; выполнен анализ пользовательских требований к разрабатываемому программному средству; составлена спецификация функциональных требований к программному средству; представлены *UML* диаграммы, отражающие поведение и аспекты взаимодействия объектов системы; представлены макеты пользовательского интерфейса; спроектирована архитектура программного средства; разработано и протестировано программное средство; составлено руководство по развертыванию и использованию программного средства.

*Область применения результатов*: могут быть использованы для визовой поддержки иностранных студентов в отделе клиентского сопровождения университета

ПЛАН ЗАДАНИЯ

БЛАНК ЗАДАНИЯ

**СОДЕРЖАНИЕ**

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc103895644)

[1 Анализ и моделирование предметной области программного средства 7](#_Toc103895645)

[2.1 Описание и анализ предметной области и программных решений 7](#_Toc103895646)

[2.2 Моделирование бизнес-процессов предметной области в нотациях *IDEF*0 и *BPMN* 8](#_Toc103895647)

[2.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 12](#_Toc103895648)

[2.4 Разработка информационной модели предметной области и ее описание 14](#_Toc103895649)

[2.5 Моделирование поведения объектов программного средства 18](#_Toc103895650)

[2.6 Моделирование взаимодействия объектов программного средства 20](#_Toc103895651)

[2.7 Постановка задачи на разработку с описанием назначения и целей создания программного средства 21](#_Toc103895652)

[3 Проектирование и разработка программного средства 22](#_Toc103895653)

[3.1 Обоснование выбора компонентов и технологий для реализации программного средства 22](#_Toc103895654)

[3.2 Проектирование пользовательского интерфейса 24](#_Toc103895655)

[3.3 Описание и разработка алгоритмов, реализующих бизнес-логику разрабатываемого программного средства 30](#_Toc103895656)

[4 Тестирование и проверка работоспособности программного средства 31](#_Toc103895657)

[5 Руководство по развертыванию и использованию программного   
средства 36](#_Toc103895658)

[7 Технико-экономическое обоснование эффективности разработки и использования программного средства 37](#_Toc103895659)

[7.1 Характеристика программного средства 37](#_Toc103895660)

[7.2 Расчет сметы затрат на разработку программного средства 37](#_Toc103895661)

[7.3 Расчет оценки уровня качества результата 41](#_Toc103895662)

[заключение 44](#_Toc103895663)

[Список использованных источников 45](#_Toc103895664)

[Приложение А (обязательное) Отчет о проверке на заимствования в   
 программном средстве «Антиплагиат» 46](#_Toc103895665)

[Приложение Б (обязательное) Листинг кода алгоритмов, реализующих   
 бизнес-логику 47](#_Toc103895666)

[Приложение В (обязательное) Листинг скрипта генерации базы данных 54](#_Toc103895667)

[Ведомость курсовой работы 58](#_Toc103895668)

# ВВЕДЕНИЕ

На сегодняшний день, для въезда или выезда из какой-либо страны, каждому физическому лицу нужна виза. Visa - разрешительный документ, дающий право человеку на пересечение тех или иных границ. Как правило, под визойпонимается фактическое разрешение иностранцу на въезд на территорию другого государства. Также эта тема не пройдет мимо иностранного студента.

Любой иностранный студент, прибывший на территорию Республики Беларусь, должен оформить студенческую визу. Каждому иностранному студенту приходится посещать многие учреждения, простояв в немалых очередях.

Целью данной курсовой работы является проектирование и разработка программного средства оптимизации процесса визовой поддержки иностранных студентов в отделе клиентского сопровождения университета.

Данная система привлечет к себе внимание многих иностранных студентов. Так как будет занимать меньше времени, меньше денежных затрат на транспорт.

В основу проектирования и реализации технических решений положены принципы исследования, анализа и оптимизации трудностей, возникающих на каждом этапе.

В следующих разделах будут подробно описаны этапы работы над курсовым проектом:

* анализ и описание предметной области для представления её существующего состояния, исследования системных требований и проблем, определения границ проекта;
* разработка функциональной модели предметной области для нахождения слабых мест существующих и рассмотрения преимуществ новых бизнес-процессов.

Курсовая работа выполнена самостоятельно, проверена в системе «Антиплагиат». Процент оригинальности составляет 90.27%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников».

Скриншот приведен в приложении А.

# АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА

## Описание и анализ предметной области и программных решений

Иностранные абитуриенты, прибывающие на обучение, получают учебную визу в посольствах и консульствах Республики Беларусь. В случае, если в стране отсутствует дипломатическая миссия Республики Беларусь, визу можно получить в «Генеральном консульском управлении «Национального аэропорта Минск».

Придя в посольство, при себе нужно иметь следующие документы:

* действительный паспорт;
* документы о среднем образовании, легализованные в установленном порядке (апостиль или легализация);
* заполненную визовую анкету;
* фото размером 35х45 (1 шт.);
* оригинал приглашения на обучение с действительным сроком въезда;
* медицинское заключение о полном обследовании, свидетельствующее о том, что кандидат физически и психически здоров и может проходить обучение в климатических условиях Республики Беларусь.

Указанный перечень документов необходимо отправить в визовый центр Республики Беларусь. Документы находятся на рассмотрении в визовом центре в течение 15 дней. В случае, если документы соответствуют всем требованиям, виза продляется на 1 год.

Также стоит помнить, что все документы, исполненные на иностранном языке, должны быть переведены на русский язык и заверены   
нотариально.

Выше представлен стандартный список документов, но требования к документам на получение визы могут быть различными для разных стран.

Иностранным гражданам, желающим обучаться в учреждении образования, необходимо получить приглашение, которое позволит открыть визу на обучение в консульском учреждении Республики Беларусь.

Чтобы получить официальное приглашение, иностранный гражданин должен предоставить в учреждение образования следующие документы:

* копию документа об образовании с указанием изученных предметов и полученных по ним на экзаменах оценок (баллов);
* копию паспорта или другого документа, удостоверяющего личность;
* заявление на обучение (с указанием паспортных данных, домашнего адреса, телефона/факса, e-mail адреса);
* медицинское заключение о состоянии здоровья и сертификат об отсутствии ВИЧ-инфекции, выданные официальным органом здравоохранения страны, из которой прибыл кандидат на учебу.

Приглашение на обучение предоставляется кандидату бесплатно.

К документам, исполненным на иностранном языке, одновременно прилагается их перевод на белорусский или русский язык, удостоверенный в нотариальном порядке.

После получения документов кандидата на обучение, учреждение образования проводит их экспертизу и принимает решение о возможности оформления приглашения на обучение.

Готовое приглашение на обучение учреждение направляет кандидату на обучение или в Посольство РБ в стране проживания кандидата.

Кандидат на обучение оформляет учебную визу в консульском учреждении Республики Беларусь по месту своего жительства.

По истечении студенческой визы, иностранный студент должен подать заявку на продление визы онлайн. Для этого, ему необходимо собрать следующие документы:

* копия паспорта;
* международное заявление;
* фото.

Сам процесс получения визы кажется быстрым и легким. Но на самом деле это занимает достаточно много времени. Не многие иностранные студенты владеют русским языком, и им бывает трудно найти дорогу или же человека, который сможет им помочь. Этим студентам приходится ходить с онлайн переводчиком, либо нанимать человека, который одновременно переведет и укажет нужный путь.

Учитывая все эти факторы, необходимо создать такой программный продукт, который облегчит работу сотрудникам визового центра, а также самим студентам.

## Моделирование бизнес-процессов предметной области в нотациях IDEF0 и BPMN

Основным бизнес-процессом системы является оказание визовой поддержки иностранных студентов. Контекстная диаграмма процесса представлена на рисунке 1.

Под визой подразумевается один заранее оговоренный на определенное время и дату. Входными параметрами (стрелка входящая слева) основного процесса являются:

* копия паспорта и фото;
* заявление.

В результате процесса получаем назначенную или отмененную визу.

При несоблюдении определенных условий, например, не правильная подача документов. Если данные условия не выполняются, то виза будет отмененной.

К «управлению» (стрелка, входящая сверху) относятся правила   
общения.

К элементам «механизма» относятся:

* сотрудник визового центра;
* руководитель визового центра.

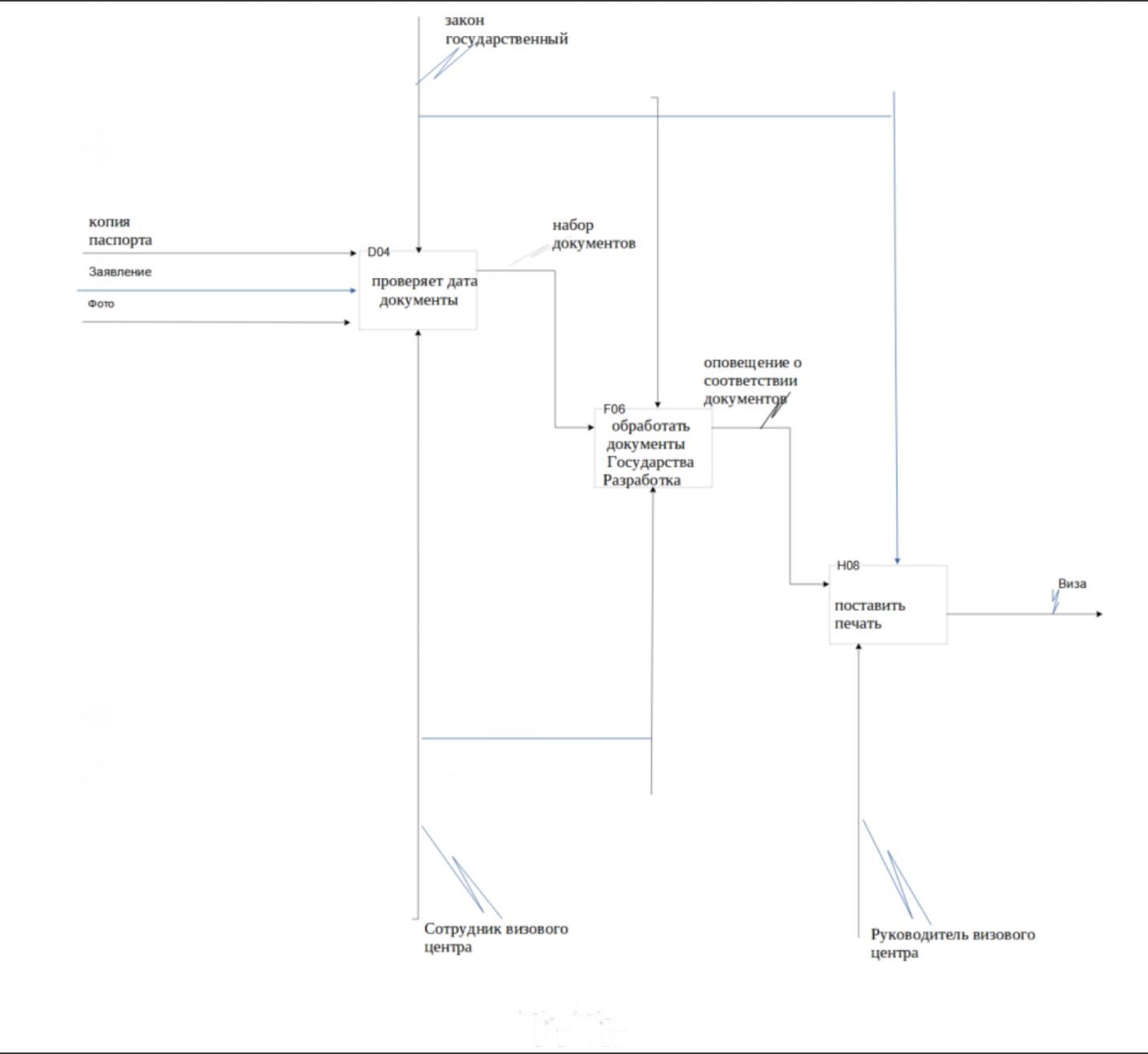


Рисунок 1 – Контекстная диаграмма *AS-IS*

Каждый из механизмов является исполнителем определённого этапа основного бизнес-процесса. Для описания уровней была разработана декомпозиция первого уровня (рисунок 2).

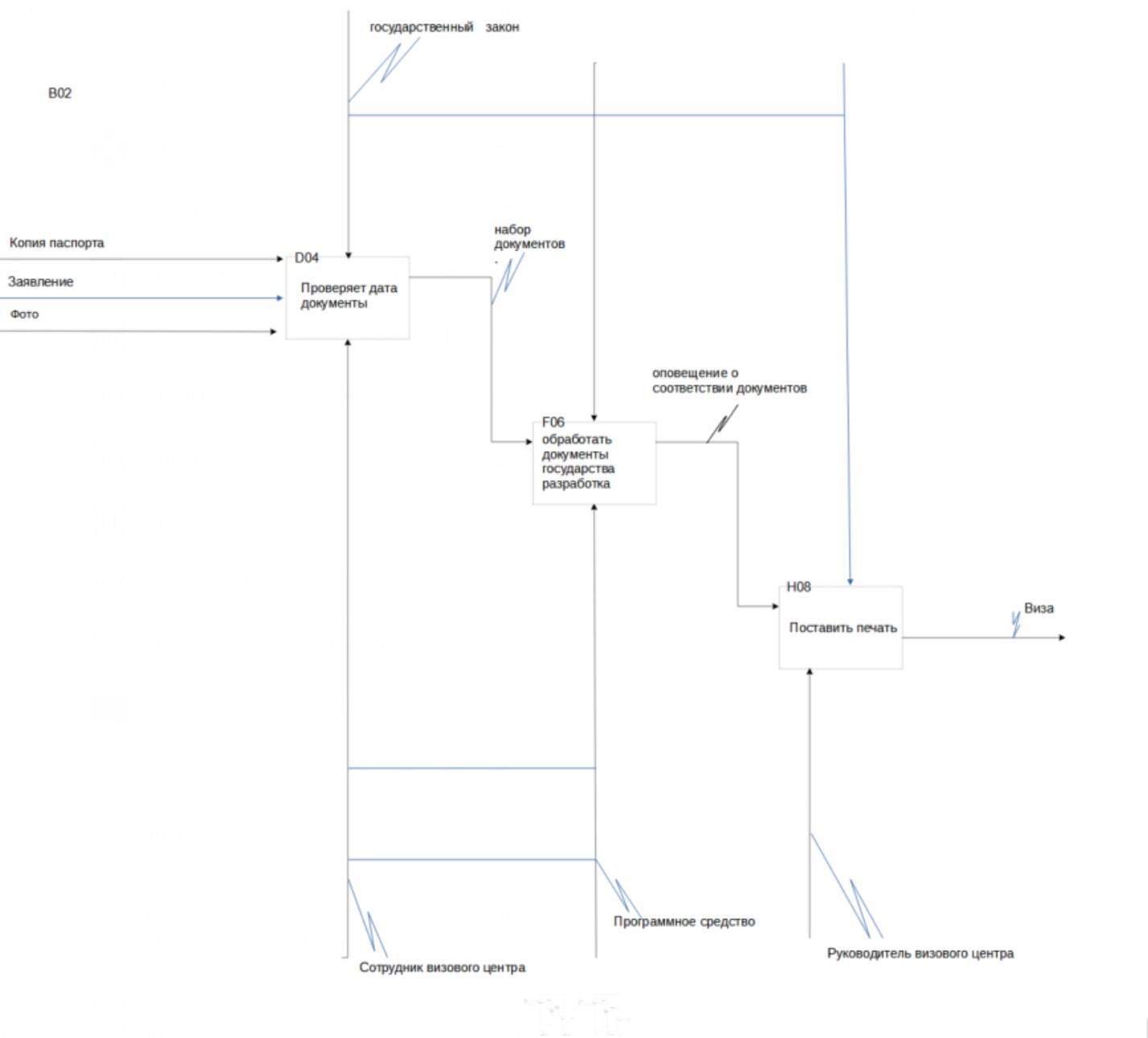


Рисунок 2 – Декомпозиция первого уровня *TO-BE*

Основными этапами в назначении встречи являются:

* копия паспорта заявления фото;
* проверка даты документов;
* набор документов;
* разработка документов государственным учреждением;
* поставить печать визы.

Цель процесса составления описания встречи заключается в подготовке получения визы и описания набора документов, основании входных параметров: дата и время, место. Это необходимо для того, чтобы обязательно донести всю самую важную информацию потенциальным студентам как можно быстрее и качественнее, не тратя время на дополнительные вопросы.

Используя подготовленное описание, происходит процесс приглашения предполагаемых студентов. Сотрудник визового центра различными методами связывается с потенциальными студентами и сообщает о получении визы либо отказе и отвечает на возможные вопросы.

Цель процесса получения и обдумывания приглашения заключается в принятии решения потенциальным органам служб, на основании которого данный центр решает одобрить в получении иностранным студентам. В результате формируется ответ на одобрение виз.

До определенного времени студент должен выполнить ряд условий визового центра, подать документы на визу, после сотрудник центра проверяет документы и принимает решение либо в отказе приемов документов, либо в их принятии, после этого сообщает ответ иностранному гражданину.

На основании условий визового центра, которые могут повлиять на утверждение документов, принимается решение. Отменена либо назначена. После данная контекстная диаграмма примет обновленный вид

Входные параметры остаются прежними, как и возможный результат. Общий процесс представлен на рисунке 3.

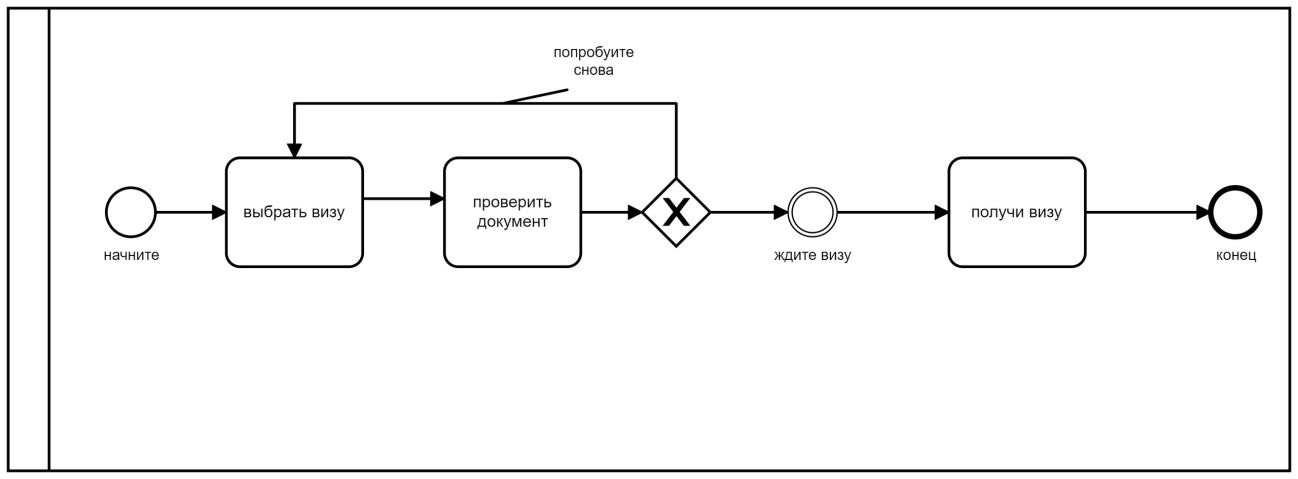


Рисунок 3 – Представление общего процесса получения визы

Основным бизнес-процессом системы является получение визы при помощи визовой поддержки. Контекстная диаграмма данного процесса представлена на рисунке.

Под визовой поддержкой подразумевается сам процесс получения учебной визы. Входными параметрами основного процесса   
являются (рисунок 4):

* фото;
* копия паспорта;
* заявление.

В результате процесса получаем учебную визу.

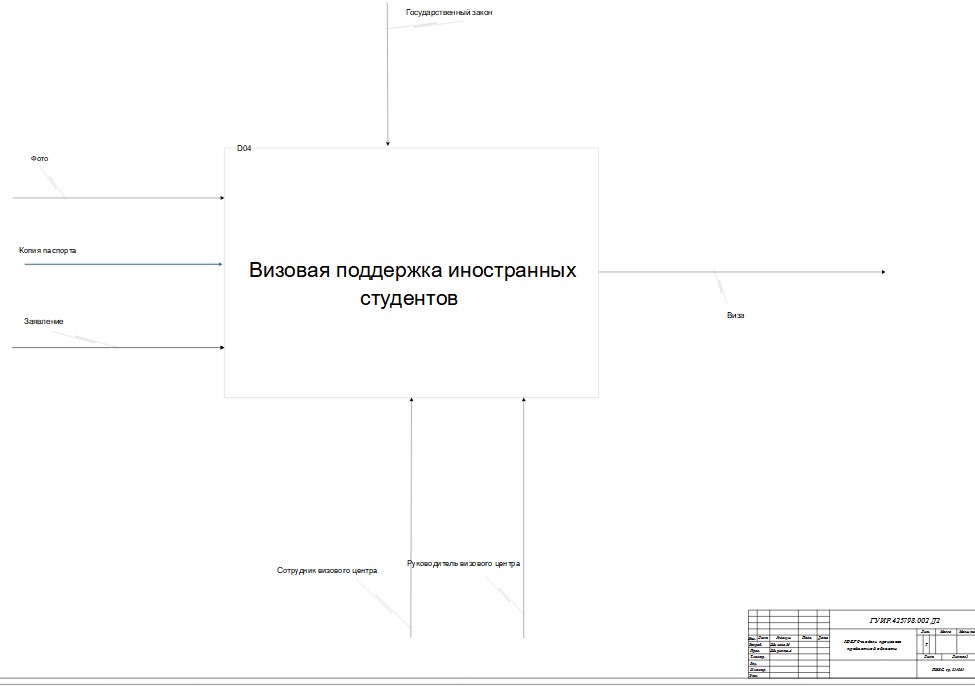


Рисунок 4 – Визовая поддержка иностранных студентов

Основными этапами в получении учебной визы являются:

* сдать заявление;
* отправить фото;
* предоставить копию паспорта.

Иностранный студент заходит в приложение и начинает регистрироваться в нём, указывая свою страну. Во-первых, во время регистрации студент вводит свою электронную почту, пароль и номер телефона. Во-вторых, после окончания регистрации ему необходимо ввести логин и пароль. В-третьих, он начинает поэтапно указывать имя и фамилию. Также необходимо ввести паспортные данные. Дальше прикрепить свою фотографию. Последним шагом является ожидание готовой визы в течение 15 дней. И на этом весь процесс получения учебной визы заканчивается.

## Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований

Функциональные требования описывают поведение системы.

Программный продукт призван помочь иностранным студентам облегчить процесс обновления визы. Предоставляет возможность удаленной подачи. Основной формат использования:

* прохождение регистрации и получение учетной записи;
* сбор документов и ввод в приложение;
* отправка документов на рассмотрение (документы получены международным отделом университета и переданы далее в обработку);
* получение ответа.

Таким образом сократится время подачи и взаимодействия между сотрудниками и подающими. В рамках приложения также можно будет просмотреть список необходимых документов и заполнить формы.

Приложение является браузерным. Работает с браузерами: *Google* *Chrome*, *Yandex* *browser*. Языковая поддержка русский, английский языки.

Для входа в приложение требуется регистрация. Данные для регистрации:

* *e-mail*;
* номер паспорта;
* номер телефона;
* пароль.

Аккаунт пользователя хранится в системе на протяжении 5 лет (время обучения).

Для подачи данных на получение визы пользователю необходимо предоставить информацию:

* копия паспорта;
* международное заявление;
* фото.

Пользователь может сохранять и редактировать информацию до момента отправки заявления на рассмотрение.

После подачи заявление фиксируется в международном отделе и берется в работу. На стороне пользователя есть возможность слежения за статусом обработки заявления.

После получения визы в течение 15 дней в базе данных хранится визовая информация. Далее удаляется.

Информация, введенная пользователем ране, не сохраняется для при последующей подаче и требует введение заново.

Система реализована на платформе *Java EE*. Используемая СУБД –*Postgres*.

На диаграмме вариантов использования (рисунок 5) выделены следующие роли: пользователь и сотрудник визового центра.

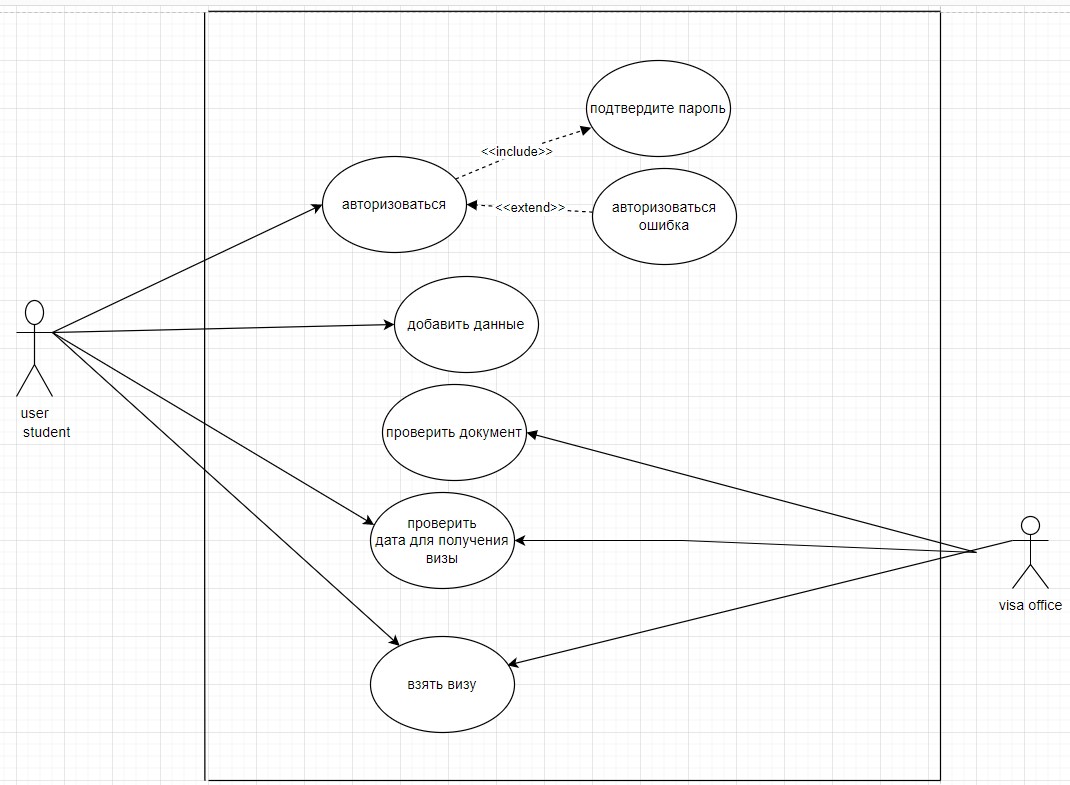


Рисунок 5 – Диаграмма вариантов использования

Если пользователь входит в систему под ролью сотрудника визового центра, то ему доступны следующие функции:

* просматривать, проверять поданные документы;
* выдавать отказ в визе на основе решения визового центра;
* подтверждать получение визы.

Если пользователь входит в систему под ролью пользователя (студента) то ему доступны следующие функции:

* регистрация, авторизация;
* подача документов и заполнение заявлений.

Разработанная диаграмма вариантов использования показывает, как будет работать разрабатываемое программное средство оптимизации процесса визовой поддержки иностранных студентов в отделе клиентского сопровождения университета.

## Разработка информационной модели предметной области и ее описание

Концептуальная схема базы данных данного модуля (рисунок 6) содержит семь основных сущностей и связи между ними.

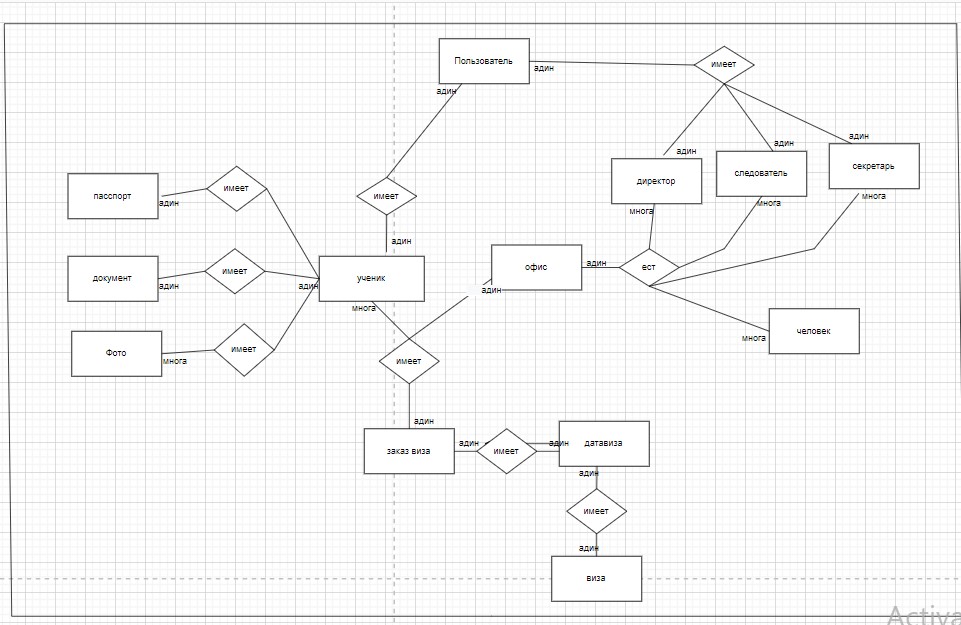


Рисунок 6 – Концептуальная схема базы данных

Под подачей визы подразумевается заранее назначенный на определенное время и дату список документов. Этапы визовой поддержки для иностранных студентов могут состоять из:

* получение официального приглашения для визы;
* срок получения приглашения: от 30 до 45 дней;
* студенту-иностранцу выдается приглашение на однократную визу сроком не более 90 дней.

Физическая схема базы данных (рисунок 7) составлена под реляционную СУБД *MySQL* в средстве визуального проектирования *MySQL Workbench*. Данная схема расширяет логическую, устанавливая типы данных и прочие параметры для атрибутов: ключ, возможно ли нулевое значение, обязательно ли уникальное значение, значение по умолчанию и т. д.

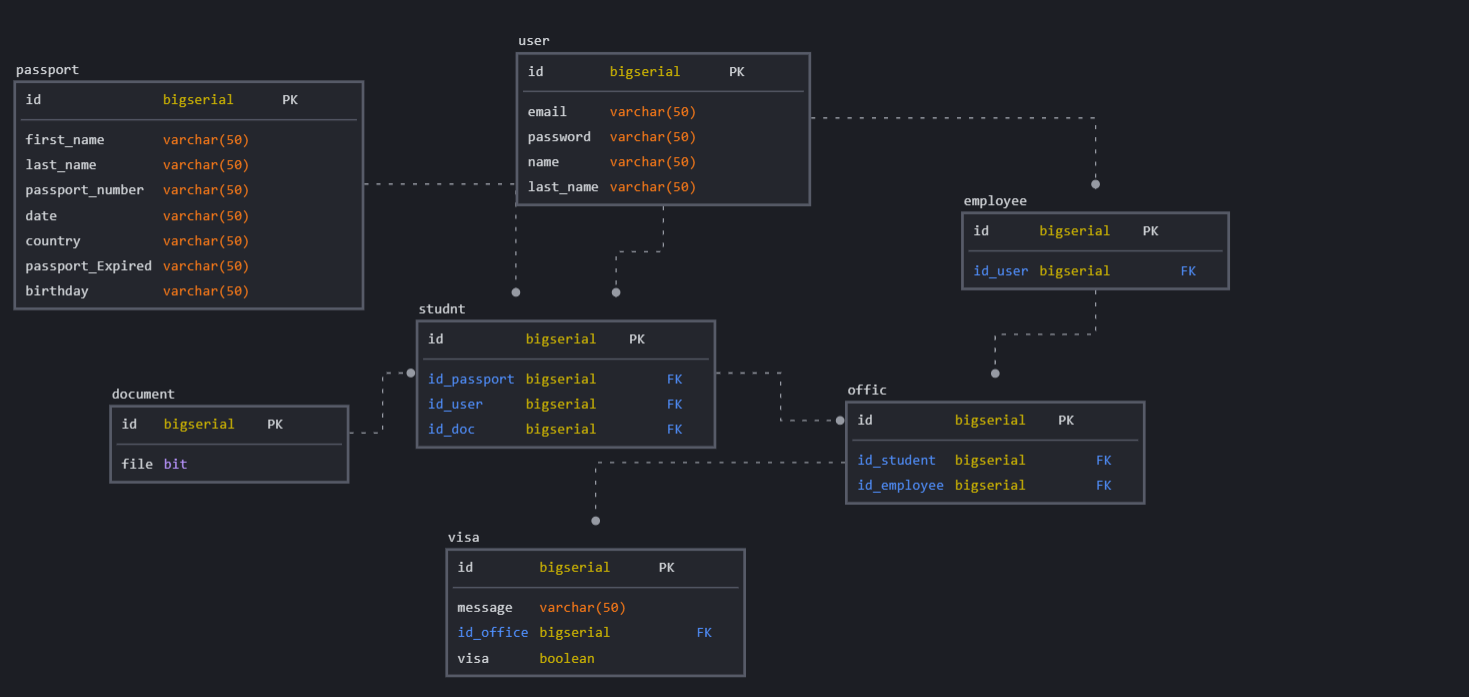


Рисунок 7 – Физическая схема базы данных

Таблица *document* используется для связи данных документов и студента и имеет следующие поля:

* *photo*. Данное поле является полем со значением типа *varchar*. Используется для хранения фото студента.
* *order\_university*. Данное поле является ключом.

Таблица *passport* используется для хранения паспортных данных и имеет следующие поля:

* *name*. Данное поле является полем со значением типа *varchar*. Оно хранит имя согласно паспортным данным.
* *last\_name*. Данное поле является полем со значением типа *varchar*. Оно хранит фамилию согласно паспортным данным.
* *middle\_name*. Данное поле является полем со значением типа *varchar*. Оно хранит отчество (среднее имя) согласно паспортным данным.
* *passport\_number.* Данное поле является полем со значением типа *varchar*. Оно хранит номер паспорта.
* *birthday*. Данное поле является полем со значением типа *varchar*. Оно хранит дату рождения.
* *expiry\_data*. Данное поле является полем со значением типа *varchar*. Оно хранит срок действия паспорта.
* *country*. Данное поле является полем со значением типа *varchar*. Оно хранит страну проживания согласно паспортным данным.
* *living\_place*. Данное поле является полем со значением типа *varchar*.

Оно хранит адрес согласно паспортным данным.

Таблица *visa* используется как ключ к визовым данным и имеет следующие поля:

* *id\_visa.* Данное поле является ключом для доступа к визовым данным.

Таблица *order* используется для связи между студентами и визовыми центрами и имеет следующие поля:

* *id\_student*. Ключ на таблицу *student*.
* *id\_office*. Ключ на таблицу *office*.

Таблица *data\_visa* используется для хранения визовых данных и имеет следующие поля:

* *id\_order*. Ключ на таблицу *order*.
* data\_order*.* Ключ на таблицу *order*.
* *data\_take.* Данное поле является полем со значением типа *date*. Оно хранит дату получения визы.
* *status\_visa*. Данное поле является полем со значением типа *boolean*.

Оно хранит статус визового заявления.

Таблица *office* используется для агрегации пользователей визового центра и имеет следущие поля:

* *id\_director.* Ключ на таблицу *director*.
* *id\_investigator.* Ключ на таблицу *investigator*.
* *secretarya.* Ключ на таблицу *secretarya*.

Таблица *student* используется для вызова и хранения информации о пользователе с ролью студент (подающие заявление на визу) и имеет следующие поля:

* *id\_passport*. Ключ на таблицу *passport*, которая содержит паспортные данные студента.
* *id\_document*. Ключ на таблицу *document*, которая содержит информацию по документам студента.
* *email*. Данное поле является полем со значением типа *varchar*. Оно хранит адрес электронной почты студента.

Таблица *director* используется для вызова информации о пользователе с ролью директор и имеет следующие поля:

* *id\_person*. Ключ на таблицу *person*.

Таблица *secretarya* используется для вызова информации о пользователе с ролью секретарь и имеет следующие поля:

* *id\_person*. Ключ на таблицу *person*.

Таблица *investigator* используется для вызова информации о пользователе с ролью investigator (проверяющий документы сотрудник) и имеет следующие поля:

* *id\_person*. Ключ на таблицу *person*.

Таблица *person* используется для хранения информации о пользователях (сотрудниках визового центра) и имеет следующие поля:

* *name*. Данное поле является полем со значением типа *varchar*. Оно хранит имя сотрудника визового центра.
* *role\_id*. Ключ на таблицу *role*.
* *user\_id.* Ключ на таблицу *users*.
* *last\_name.* Данное поле является полем со значением типа *varchar*. Оно хранит фамилию сотрудника визового центра.
* *phone\_number*. Данное поле является полем со значением типа *varchar*. Оно хранит номер телефона сотрудника визового центра.
* *email*. Данное поле является полем со значением типа *varchar*. Оно хранит адрес электронной почты сотрудника визового центра.

Таблица *users* используется для хранения данных пользователей, необходимых для входа в систему, и имеет следующие поля:

* *login*. Данное поле является полем со значением типа *varchar*. Оно содержит имя входа пользователя, указанное при регистрации.
* *password*. Данное поле является полем со значением типа *varchar*.

Оно содержит пароль пользователя, заданный при регистрации.

Таблица *role* используется для хранения ролей пользователей и имеет следующие поля:

* *rolename*. Данное поле является полем со значением типа *varchar*. Оно содержит наименование роли.

Информационная модель приведена к третьей нормальной форме и позволяет смоделировать логику программного средства, наполнить его данными, обеспечить их целостность, возможность распределенного и одновременного доступа к информации.

Проектирование информационной модели является важным этапом моделирования предметной области, после осуществления которого следует этап моделирования внутренних особенностей будущего программного средства.

## Моделирование поведения объектов программного средства

В практическом отношении, для создания представления программного средства используются *UML* диаграммы. На английском *UML* – *Unified Modeling Language*, в переводе на русский – унифицированный язык моделирования. С его помощью можно изображать организационные структуры, моделировать бизнес-процессы, проектировать программное средство.

*UML* является открытым стандартом, помогает описать абстрактную модель программного средства, используется для визуализации, документирования и проектирования. На основании *UML* диаграмм можно сгенерировать код.

Примерами *UML* диаграмм могут быть диаграммы:

* классов;
* компонентов;
* развёртывания;
* состояний;
* последовательности;
* вариантов использования;
* коммуникации.

Первые 6 диаграмм отражают и описывают представление модели программного средства [1].

Для обозначения того, в каких состояниях может находиться иностранная виза была разработана диаграмма состояний (рисунок 8). На данной диаграмме можно увидеть все возможные состояния, а также переходы между ними. Состояния встречи необходимы для информирования участников, на какой стадии находится та или иная встреча.

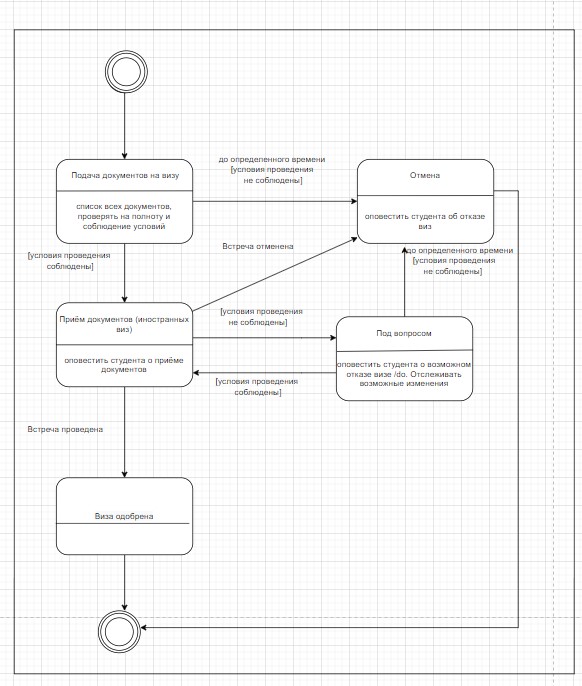


Рисунок 8 – Диаграмма состояний визы

Процесс изменения состояний начинается с подачи документов на визу. По умолчанию, после создания, новая встреча имеет состояние «подача документов на визу».

Целью получения любой визы является непосредственно её проведение (состояние «одобрено»), до которого участники будут получать уведомления обо всех изменениях в состоянии встречи. До достижения состояния «Проведена» есть минимум еще одно состояние, которое должно быть пройдено – «прием документов».

Состояние «прием документов» достигается по соблюдению условий. Важным условием достижения данного состояния является необходимость уложиться к запанированному времени.

После назначение встречи, выполнение условий проведения может измениться, например, несколько участников изменят свой статус на «Не приду». Для таких случаев предусмотрено ещё одно состояние – «Под вопросом». Данное состояние информирует участников, что условия не выполняются и виза не может быть, одобрена из-за изменений в соблюдении условий.

После истечения запланированного времени, если условия изменений были не соблюдены то, состояние переводиться в «Отменена», которое информирует участников об отмене визы. Процесс организации заканчивается.

Виза может быть переведена в состояние «Отменена» в любой момент сотрудником визового центра, если она не является проведенной.

## Моделирование взаимодействия объектов программного средства

Для демонстрации работы аутентификации в системе была разработана диаграмма последовательности (рисунок 9). На ней изображено взаимодействие клиентской части приложения с микросервисной архитектурой серверной части.

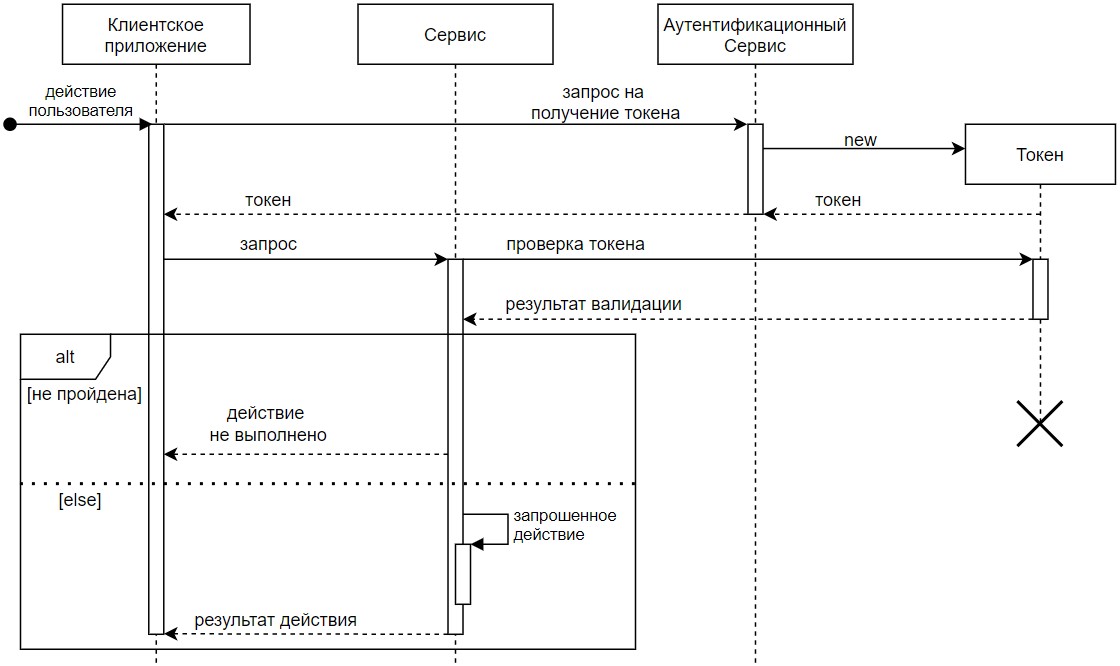


Рисунок 9 – Диаграмма последовательности аутентификации

Для того, чтобы на сервере было выполнено любое действие, инициируемое пользователем при взаимодействии с клиентской частью, необходимо, чтобы данный пользователей был аутентифицирован (имел валидный токен).

За процесс аутентификации пользователей отвечает специальный аутентификационный сервис.

Последовательность действий, следующая:

* пользователь инициирует действие на клиентской части;
* клиентская часть делает запрос в аутентификационный сервис, для подтверждения личности пользователя;
* аутентификационный сервис, на основании данных о пользователе, подписывает новый токен;
* аутентификационный сервис возвращает в ответе клиенту токен;
* клиент делает запрос на необходимый сервис, для обработки инициированного пользователем действия, вместе с полученным от аутентификационного сервиса токеном;
* сервис проверяет данный токен на валидность.

В зависимости от результатов: валиден токен или нет, сервис выполняет или не выполняет действие и возвращает клиенту соответствующий ответ.

## Постановка задачи на разработку с описанием назначения и целей создания программного средства

**Назначение.** Конечный продукт – многофункциональный сервис для организации и поддержки иностранным студентам визы, помогающий упростить процессы, с которыми постоянно сталкиваются любые иностранные студенты

Система является альтернативой другим более узкоспециализированным приложениям, которые не поддерживают большого количества дисциплин, популяризирую менее известные из них.

**Цель создания.** Система автоматизирует процессы, связанные с поиском и организацией поддержки виз иностранцам.

В процессе организации система может выступать централизованным решением.

Система помогает обеспечить экономию времени и денег.

# ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## Обоснование выбора компонентов и технологий для реализации программного средства

В качестве основного языка разработки был выбран объектно-ориентированный язык *Java*, так как он позволяет разрабатывать приложения независимо от конечной пользовательской архитектуры. Это обусловлено тем, что для выполнения кода приложения на *Java* требуется лишь *JVM* (*Java Virtual Machine* – виртуальная машина *Java*). Байт-код, который получается в результате компиляции приложений написанных на Java, может быть запущен везде именно благодаря виртуальной машине. Данный факт обеспечивает такое качество приложений, как кроссплатформенность.

*Java* создавался как язык для распределенного программирования: он имеет встроенный механизм совместного использования данных и программ несколькими компьютерами, что повышает производительность и эффективность труда.

Другой важной особенностью технологии *Java* является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Для реализации полиморфизма в *Java* не требуется указание дополнительных модификаторов, поскольку *Java* поддерживает полиморфизм по умолчанию

Идея пространств имён воплощена в *Java*-пакетах. Внутри пакета есть два независимых пространства имен: переменные и методы [2].

Протокол *TCP*/*IP* – сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю. В модели предполагается прохождение информации через четыре уровня, каждый из которых описывается правилом (протоколом передачи). Наборы правил, решающих задачу по передаче данных, составляют стек протоколов передачи данных, на которых базируется Интернет.

Набор интернет-протоколов обеспечивает сквозную передачу данных, определяющую, как данные должны пакетироваться, обрабатываться, передаваться, маршрутизироваться и приниматься. Эта функциональность организована в четыре слоя абстракции, которые классифицируют все связанные протоколы в соответствии с объемом задействованных сетей. От самого низкого до самого высокого уровня – это уровень связи, содержащий методы связи для данных, которые остаются в пределах одного сегмента сети (ссылка); интернет-уровень, обеспечивающий межсетевое взаимодействие между независимыми сетями; транспортный уровень, обрабатывающий связь между хостами; и прикладной уровень, который обеспечивает обмен данными между процессами для приложений.

Стек протоколов *TCP*/*IP* включает в себя четыре уровня:

* прикладной уровень (*Application Layer*);
* транспортный уровень (*Transport Layer*);
* межсетевой уровень (Сетевой уровень) (*Internet Layer*); − канальный уровень (*Network Access Layer*).

Протоколы этих уровней полностью реализуют функциональные возможности модели *OSI*. На стеке протоколов *TCP*/*IP* построено всё взаимодействие пользователей в *IP*-сетях. Стек является независимым от физической среды передачи данных, благодаря чему, в частности, обеспечивается полностью прозрачное взаимодействие между проводными и беспроводными сетями [3].

PostgreSQL – свободная объектно-реляционная система управления базами данных.

Сильными сторонами PostgreSQL считаются:

* высокопроизводительные и надёжные механизмы транзакций и репликации;
* расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются *PL/pgSQL, PL/Perl, PL/Python* и *PL/Tcl*; дополнительно можно использовать *PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh* и *PL/V*8, а также имеется поддержка загрузки модулей расширения на языке C
* наследование;
* возможность индексирования геометрических (в частности, географических) объектов и наличие базирующегося на ней расширения *PostGIS*;
* встроенная поддержка слабоструктурированных данных в формате *JSON* с возможностью их индексации;
* расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных) [4].

Текущие ограничения PostgreSQL:

* нет ограничений на максимальный размер базы данных;
* нет ограничений на количество записей в таблице;
* нет ограничений на количество индексов в таблице;
* мксимальный размер таблицы – 32 Тбайт;
* максимальный размер записи – 1,6 Тбайт;
* максимальный размер поля – 1 Гбайт;
* максимум полей в записи250—1600 (в зависимости от типов полей).

Особенности PostgreSQL:

1. Функции в PostgreSQL являются блоками кода, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки собственно SQL и требует использования некоторых языковых расширений. Функции могут писаться с использованием различных языков программирования. PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса. Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя. Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть различие.
2. Триггеры в PostgreSQL определяются как функции, инициируемые DML-операциями. Например, операция INSERT может запускать триггер, проверяющий добавленную запись на соответствия определённым условиям. При написании функций для триггеров могут использоваться различные языки программирования. Триггеры ассоциируются с таблицами. Множественные триггеры выполняются в алфавитном порядке.
3. Механизм правил в PostgreSQL представляет собой механизм создания пользовательских обработчиков не только DML-операций, но и операции выборки. Основное отличие от механизма триггеров заключается в том, что правила срабатывают на этапе разбора запроса, до выбора оптимального плана выполнения и самого процесса выполнения. Правила позволяют переопределять поведение системы при выполнении SQLоперации к таблице.
4. Многоверсионность поддерживается в PostgreSQL — возможна одновременная модификация БД несколькими пользователями.
5. Расширение PostgreSQL для собственных нужд возможно практически в любом аспекте. Есть возможность добавлять собственные преобразования типов, типы данных, домены (пользовательские типы с изначально наложенными ограничениями), функции (включая агрегатные), индексы, операторы (включая переопределение уже существующих) и процедурные языки.
6. Наследование в PostgreSQL реализовано на уровне таблиц. Таблицы могут наследовать характеристики и наборы полей от других таблиц (родительских). При этом данные, добавленные в порождённую таблицу, автоматически будут участвовать (если это не указано отдельно) в запросах к родительской таблице [5].

## Проектирование пользовательского интерфейса

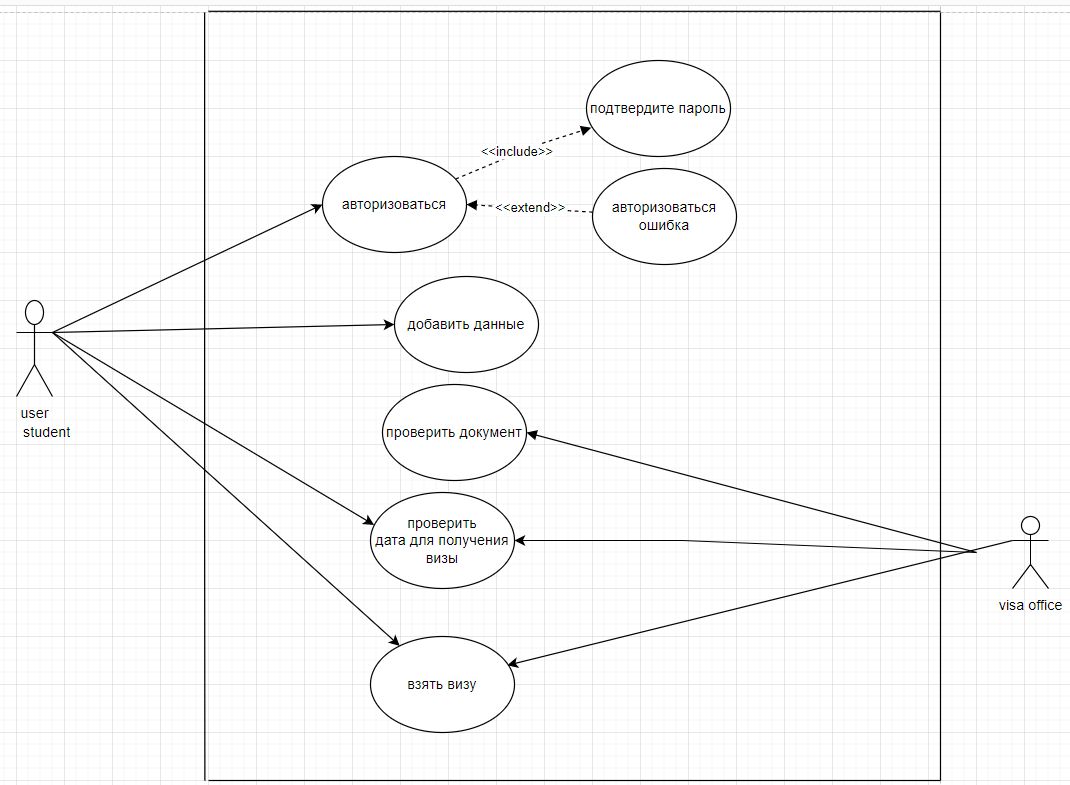


Рисунок 11 – Диаграмма вариантов использования

Если пользователь входит в систему под ролью аналитика, то ему доступны следующие функции:

* создать логин пароль;
* добавить дату;
* прием документов;
* проверка на срок даты визы;
* получение визы;
* верификация паспорта;
* ошибка введения логина;
* нет полноты документов;
* ошибки в документах;
* получение визы;
* отказ визы.

Карта пользовательских интерфейсов.

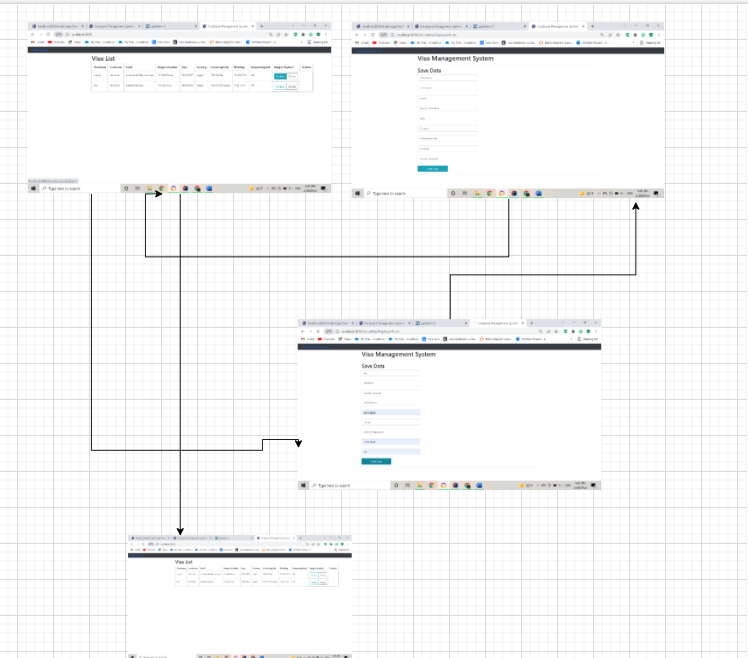


Рисунок 12 – Карта пользовательских интерфейсов.

Результаты реализации клиентской части ПС. Представлены следующие окна приложения.

Окно регистрации для сотрудника визового центра представлено на   
рисунке 12.

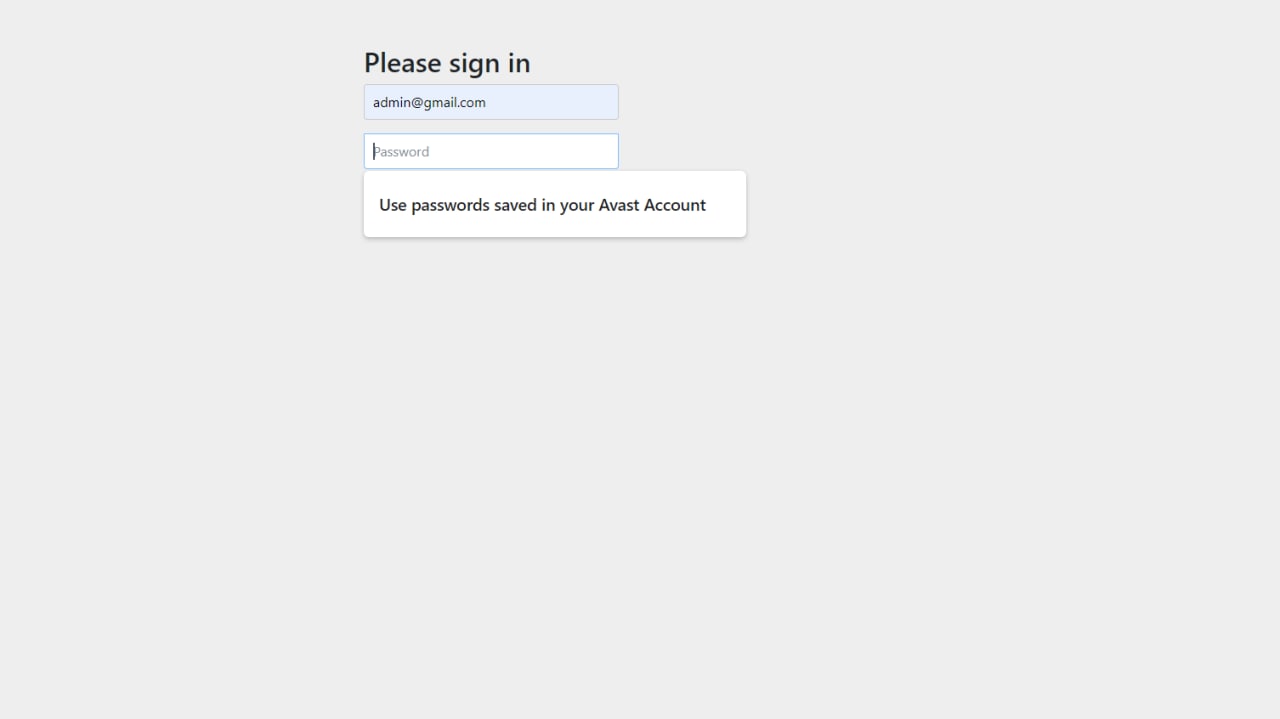


Рисунок 13 – Окно регистрации сотрудника визового центра.

Окно входа в приложение представлено на рисунке 13.

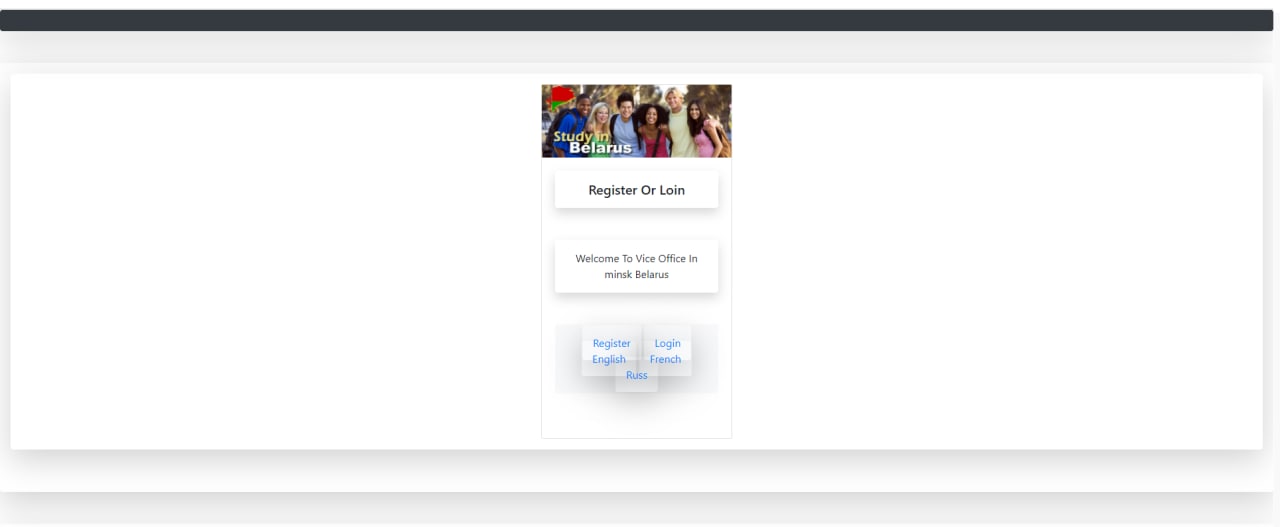


Рисунок 14 – Окно входа в приложение

Окно подачи заявления представлено на рисунке 14.

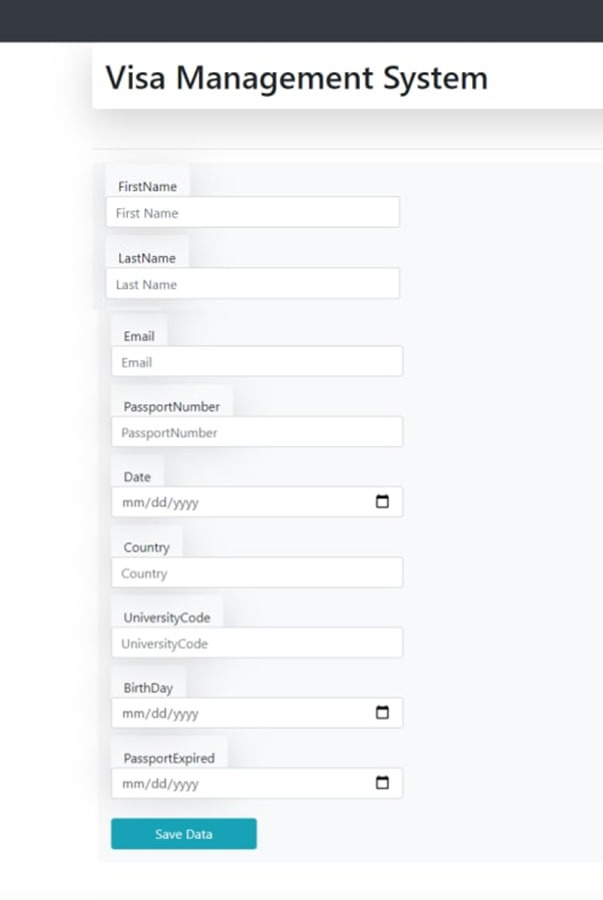


Рисунок 15 – Окно подачи визового заявления.

Окно регистрации для студента представлено на рисунке 16.

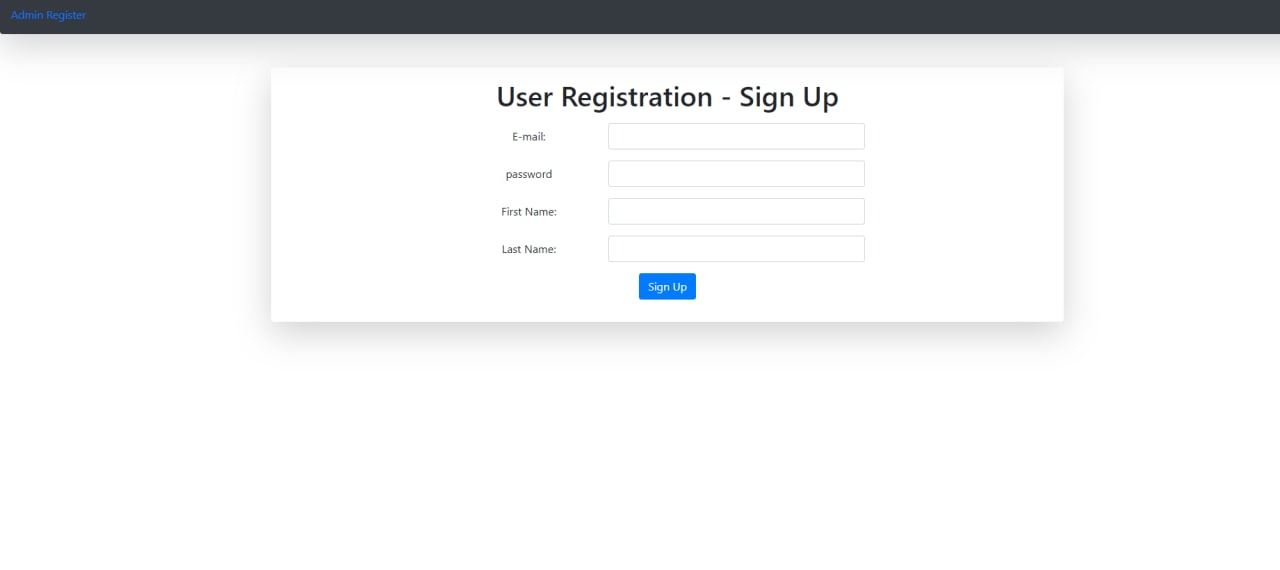


Рисунок 16 – Окно регистрации для студента.

Диаграмма компонентов, представленная на рисунке 17, позволяет определить состав программных компонентов, в роли которых может выступать исходный, бинарный и исполняемый код, а также установить зависимости между ними.

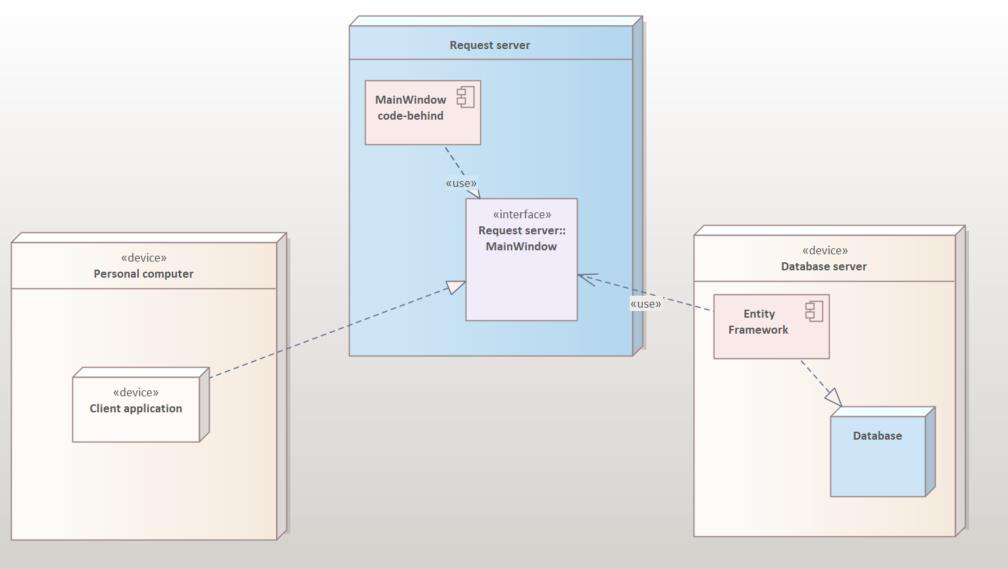


Рисунок 17 – Диаграмма компонентов

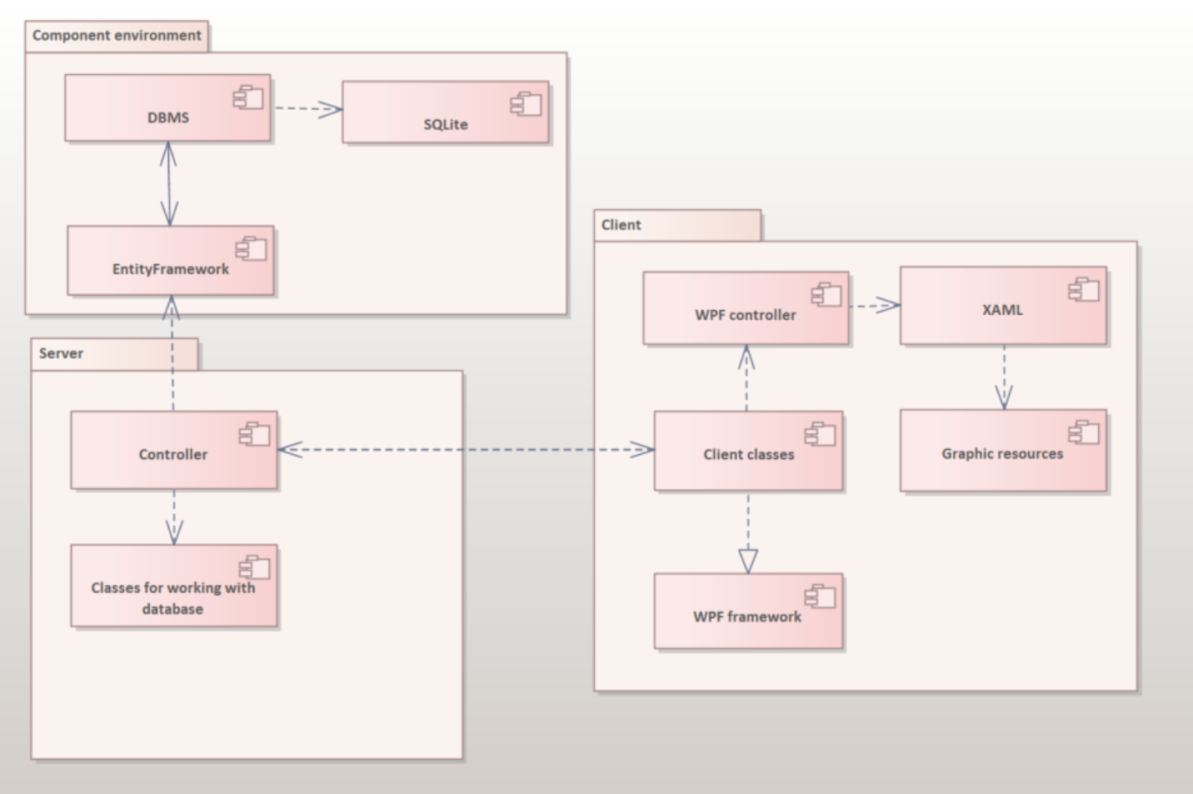


Рисунок 18 – Диаграмма развёртывания

Диаграмма развертывания позволяет решить ряд второстепенных, но важных задач, как, например, обеспечение безопасности системы.  
Среди преследуемых целей при разработке данной диаграммы (см. рисунок 18) можно выделить распределение компонентов системы по ее физическим узлам, отображение физических связей между узлами системы на этапе исполнения, а также выявление узких мест системы и реконфигурация ее топологии для достижения требуемой производительности.

## Описание и разработка алгоритмов, реализующих бизнес-логику разрабатываемого программного средства

Бизнес-логика – это реализация предметной области в информационной системе. Является синонимом термина «логика предметной области». Бизнес-логика задает правила, которым подчиняются данные предметной области.

В фазе бизнес-моделирования и разработки требований бизнес-логика может описываться в виде [6]:

* текста;
* концептуальных аналитических моделей предметной области (онтологии);
* бизнес-правил;
* разнообразных алгоритмов;
* диаграмм деятельности;
* графов и диаграмм перехода состояний;
* моделей бизнес-процессов.

В фазе анализа и проектирования системы бизнес-логика воплощается в различных диаграммах языка *UML* или ему подобных. В фазе программирования бизнес-логика воплощается в коде классов и их методов, в случае использования объектно-ориентированных языков программирования, или процедур и функций, в случае применения процедурных языков.

В многоуровневых (многослойных) информационных системах этот уровень взаимодействует с нижележащим уровнем инфраструктурных сервисов, например, интерфейсом доступа к базе данных или файловой системе и вышележащим уровнем сервисов приложения, который уже, в свою очередь, взаимодействует с уровнем пользовательского интерфейса или внешними системами.

# ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование – процесс анализа программного средства и сопутствующей документации с целью поиска дефектов и повышения качества продукта [7]. В рамках тестирования разработанного программного средства для проверки уровня базовых пользовательских требований было выполнено тестирование на основе тест-кейсов. Результаты тестирования приведены в таблице 1.

Таблица 1 – Тест-кейсы для проверки уровня базовых пользовательских   
требований

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Идентификатор** | **Заглавие тесткейса** | **Шаги тест-кейса** | **Ожидаемый результат** | **Фактический**  **результат** |
| 1 | 2 | 3 | 4 | 5 |
| *UC-1* | Выполнить регистрацию | 1 Открыть программное средство.  2 Перейти на  форму регистрации.  3 Заполнить обязательные поля на форме регистрации.  4 Подтвердить регистрацию.  5 Авторизоваться в системе, используя логин и пароль, заданные при регистрации. | Пользователь авторизован. Регистрация пройдена успешно. | Пользователь авторизован. Тест-кейс пройден. |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 |  | 3 | 4 | 5 |
| *UC-2* | Заполнить заявление документы визу | и на | 1 Авторизовать ся в системе.  2 Перейти на форму подачи визового заявления.  3 Заполнить необходимые формы, приложить документы.  4 Подтвердить подачу. | Статус заявления в прогрессе. | Заявление подано. Тесткейс пройден. |
| *UC-3* | Заполнить  заявление и документы на визу и попытаться отредактировать | | 1 Авторизоваться в системе.  2 Перейти на форму подачи визового заявления. 3 Заполнить необходимые формы, приложить документы.  4 Подтвердить подачу.  5 Открыть заявление, отредактировать. | Редактирование поданного заявления невозможно. | Пользователь  не смог  отредактировать заявление. Тест-кейс пройден. |

Окончание таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| *UC-4* | Сотрудник визового центра может открыть поданное  заявление и принять решение о выдаче визы | 1 Авторизоваться в системе в качестве сотрудника визовой службы.  2 Открыть  любое поданное заявление.  3 Одобрить выдачу визы. | Визовое заявление переходит в статус  «одобрено». | Визовое заявление перешло в корректный статус. Тесткейс пройден. |
| *UC-5* | Сотрудник визового центра может открыть поданное  заявление и принять решение об отказе в выдаче визы | 1 Авторизоваться в системе в качестве сотрудника визовой службы.  2 Открыть  любое поданное заявление.  3.Отказать в  выдаче визы | Визовое заявление переходит в статус  «отказано». | Визовое заявление  перешло в  корректный статус.  Тесткейс пройден. |
| *UC-6* | Пользователь может следить за статусом поданного заявления | 1 Авторизоваться в системе.  2 Открыть  поданное визовое заявления | В зависимости от статуса должно быть отображено: в прогрессе, отказ, одобрено | Визовое заявление содержит корректный статус.  Тесткейс пройден. |

**Модульное тестирование.**

Код тестов:

@RunWith**(**MockitoJUnitRunner.class**)**

**public** **class** ListStudentServiceTest **{**

@Mock

**private** StudentRepository studentRepository;

@InjectMocks

**private** ListStudentService listStudentService;

@Test

**public** void shouldReturnAllStudent **()** **{**

List**<**Student**>** student= new ArrayList**()**;

student.add**(**new Student **())**;

given**(**studentRepository.findAll**())**.willReturn**(**student**)**;

List**<**Student**>** expected = listStudentService.listAllStudent **()**;

assertEquals**(**expected, Student**)**;

verify**(**userRepository**)**.findAll**()**;

**}**

**}**

Таблица 2 – Результаты автоматизированного тестирования моделей

|  |  |  |
| --- | --- | --- |
| Тестируемый тип | Дата проведения теста | Результаты тестирования |
| student | 1.03.2022 | Можно получить список студентов зарегистрированных в приложении |
|  |  |  |
| office | 1.03.2022 | Изменяется (возможно удаление) |
| user | 1.03.2022 | Возможность создания нового пользователя и обновление его данных |

Текст других Unit-тестов.

require "test\_helper"

@RunWith**(**MockitoJUnitRunner.class**)**

**public** **class** DeleteOfficeServiceTest **{**

@Mock

**private** OfficeRepository OfficeRepository;

@InjectMocks

**private** DeleteOfficeService deleteOfficeService;

@Test

**public** void whenGivenId\_shouldDeleteUser\_ifFound**(){**

Office office= new Office **()**;

office.setName**(**“Test Name”**)**;

office.setId**(**1L**)**;when**(**officeRepository.findById**(**office.getId**()))**.thenReturn**(**Optional.of**(**office**))**;deleteofficeService.deleteUser**(**office.getId**())**;

verify**(**officeRepository**)**.deleteById**(**office.getId**())**;

**}**

@Test**(**expected = RuntimeException.class**)**

**public** void should\_throw\_exception\_when\_Office\_doesnt\_exist**()** **{**

Office office= new Office **()**;

office.setId**(**89L**)**;

office.setName**(**“Test Name”**)**;

given**(**userRepository.findById**(**anyLong**()))**.willReturn**(**Optional.ofNullable**(null))**;deleteUserService.deleteUser**(**user.getId**())**;

**}**

**}**

@RunWith**(**MockitoJUnitRunner.class**)**

**public** **class** UpdateUserServiceTest **{**

@Mock

**private** UserRepository userRepository;

@InjectMocks

**private** UpdateUserService updateUserService;

@Test

**public** void whenGivenId\_shouldUpdateUser\_ifFound**()** **{**

User user = new User**()**;user.setId**(**89L**)**;user.setName**(**“Test Name”**)**;

User newUser = new User**()**;

user.setName**(**“New Test Name”**)**;

given**(**userRepository.findById**(**user.getId**()))**.willReturn**(**Optional.of**(**user**))**;

updateUserService.updateUser**(**user.getId**()**, newUser**)**;

verify**(**userRepository**)**.save**(**newUser**)**;verify**(**userRepository**)**.findById**(**user.getId**())**;

**}**

@Test**(**expected = RuntimeException.class**)**

**public** void should\_throw\_exception\_when\_user\_doesnt\_exist**()** **{**

User user = new User**()**;

user.setId**(**89L**)**;user.setName**(**“Test Name”**)**;

User newUser = new User**()**;newUser.setId**(**90L**)**;

user.setName**(**“New Test Name”**)**;given**(**userRepository.findById**(**anyLong**()))**.willReturn**(**Optional.ofNullable**(null))**;updateUserService.updateUser**(**user.getId**()**, newUser**)**;

**}**

**}**

Разработанные *Unit*-тесты для проверки работоспособности кода серверной части ПС. Все тесты прошли успешно. Весь код *Unit*-тестов представлен в публичном репозитории.

# РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

**Руководство по установке программного средства.** В рамках данного раздела представлен алгоритм по установке и использованию программного средства на любом персональном компьютере.

Состоит данный раздел из трёх пунктов:

* требования к персональному компьютеру;
* развёртывание БД на сервере СУБД;
* развёртывание программного средства на вычислительном устройстве.

Что касается требований к устройству, оно должно поддерживать СУБД *Postgres*.

Естественно, все предъявляемые требования характеризуют поддержку версий технологий, используемых при создании приложения. Данные требования выполняются на большинстве обычных устройств.

Что касается второго пункта – это развёртывание БД на сервере. Его можно описать следующим алгоритмом:

1. Запустить интерфейс управления выбранной СУБД на сервере.
2. Создать схему базы данных, если она не создана, выбрать её.

Создание базы данных завершено.

Алгоритм развёртывания программного средства на устройстве представлен далее:

1. Распаковать архив с программным средством.
2. Выполнить сборку проектов для программного средства.
3. Настроить соединение с БД, которая, как предполагается, уже развёрнута в рамках второго пункта данного раздела.
4. Запустить сначала работу сервера, а затем клиентскую часть.

После этого установка программного средства завершена. Оно полностью готово к использованию и успешно функционирует.

Доступ к приложению возможен посредством *Google Chrome, Yandex browser*.

# ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА

## Характеристика программного средства

Целью дипломного проекта является создание программного средства для собственных нужд УВО БГУИР, которое представляет собой приложение для оптимизации процесса визовой поддержки иностранных студентов в отделе клиентского сопровождения университета. Т.к. программное средство предназначено для эффективности сбора и анализа информации, технико-экономическое обоснование рассматривается в рамках научно-исследовательской работы (НИР).

Функционал данного программного средства позволит иностранным студентам подавать заявку на обновление визы, прикреплять документы, а также передавать личные данные в режиме онлайн в отдел клиентского сопровождения университета. Данный функционал нацелен на снижение временных затрат.

* сметы затрат ();
* оценки уровня качества (Кк).

## Расчет сметы затрат на разработку программного средства

Расчет основной заработной платы участников команды осуществляется по формуле:

(1)

где *n*-количество исполнителей, занятых разработкой программного средства; — дневная тарифная ставка *i*-го исполнителя (руб.); —эффективный фонд рабочего времени *і*-го исполнителя (дней); *—*коэффициент премирования (1,5) для руководителя, (1,2) для исполнителя.

Базовая ставка специалиста бюджетной организации на 01.04.2022 составляет 209 руб. В настоящее время среднемесячная зарплата в организации составляет 1800 руб. Среднемесячная расчетная норма рабочего времени – 176 часов.

Расчет затрат на основную заработную плату исполнителей проекта приведен в таблице 3.

Таблица 3 - Расчёт основной заработной платы исполнителей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Должность | Месячная заработная плата с учётом надбавок, руб. | Дневная тарифная ставка, руб. | Плановый фонд рабочего времени, дн. | Основная заработная плата исполнителя, руб. |
| Руководитель  проекта | 1199,94 | 57,14 | 10 | 571,4 |
| Программист | 800,1 | 38,1 | 30 | 1143 |
| Итого  основная  заработная  плата |  |  |  | 1714,4 |
| Итого с учетом премии (ЗО) |  |  |  | 857 |
| Всего основная заработная плата |  |  |  | 2571,4 |

Затраты на дополнительную заработную плату команды определяются по формуле:

где ЗО - затраты на основную заработную плату, руб.; НД - норматив дополнительной заработной платы (10%).

После подстановки значений в формулу (2) размер дополнительной

заработной платы команды составит:

Отчисления в фонд социальной защиты населения и на обязательное страхование (ЗC3) определяется в соответствии с действующим законодательным актами по формуле:

где HСЗ – норматив отчислений в фонд социальной защиты населения и на обязательное страхование (34+0,6%).

Размер отчислений в фонд социальной защиты населения и на обязательное страхование, согласно формуле (3), составит:

Затраты машинного времени на разработку программы. Расходы по

статье «Машинное время» (РМ) включает оплату машинного времени, необходимого для разработки и отладки программного продукта и

определяется по формуле:

где Цм — цена одного машинного часа; Тч — количество часов работы в день; Cр — длительность проекта, дни.

Стоимость машинного часа составляет 0,8 руб. Разработка проекта займет 30 дней для программиста и 10 дней для руководителя проекта. Определим по формуле (4) затраты по статье «Машинное время»:

Затраты по статье «Прочие расходы» (Рп)

где Нпр – норматив прочих расходов, 50%.

После подстановки значений в формулу (5) прочие расходы при разработке системы составят:

Общая сумма расходов по всем статьям сметы (Ср) на ПО рассчитывается по формуле:

Итого, общая сумма затрат составит:

Затраты на сопровождение и адаптацию программного средства определяются по формуле:

где Нрса - норматив расходов на сопровождение и адаптацию, 20%; Ср - смета расходов в целом по организации без расходов на сопровождение и адаптацию, руб.

Затраты на сопровождение и адаптацию программного средства по формуле (7) составят:

Общая сумма расходов на разработку (с затратами на сопровождение)

как полная себестоимость программного продукта () рассчитывается по

формуле:

Полная себестоимость программного средства, рассчитанная по формуле (5.8), составит:

Далее рассчитаем плановую прибыль по формуле:

Итого плановая прибыль составит:

Сметная стоимость рассчитывается по формуле:

Итого сметная стоимость составит:

Все расчеты сметы затрат сведены в таблицу 5.

Таблица 5 – Смета затрат на разработку программного продукта

|  |  |  |
| --- | --- | --- |
| Наименование статей | Условное обозначение | Значение (руб. ) |
| 1 | 2 | 3 |
| Основная заработная плата исполнителей | ЗО | 2571,4 |
| Дополнительная заработная плата  команды разработчиков | ЗД | 257,14 |
| Отчисления в фонд социальной защиты  населения и на обязательное страхование | ЗСЗ | 882,9 |
| Машинное время | РМ | 256 |
| Прочие расходы | РП | 1285,7 |
| Общая сумма расходов по всем статьям сметы | СР | 3538,74 |
| Затраты на сопровождение и адаптацию |  | 707,748 |
| Полная себестоимость | СП | 4246, 488 |
| Плановая прибыль | П | 1273,94 |
| Сметная себестоимость |  |  |

Так как данное программное средство предназначено для собственных нужд УВО БГУИР, то мы ограничимся расчётом полной себестоимости.

## Расчет оценки уровня качества результата

Количественная оценка уровня качества результата, полученного в

ходе дипломного проекта, рассчитывается по формуле:

где ‒ комплексный показатель достигнутого уровня качества результата выполненных работ; - нормированный коэффициент значимости *і*-го критерия, используемого для оценки; - достигнутый уровень по *і*-му критерию; *n* - количество критериев прогрессивности и полезности результатов,

полученных в дипломном проекте.

Для оценки результативности НИР используют различные критерии.

Важнейшими из них являются новизна, значимость для практики, объективность, доказательность, точность. Показатели каждого критерия классифицируются по 5 уровням качества. Рассмотрим детальнее оценку и значимость данных критериев.

Новизна. Новизна определяется как наличие в результатах НИР новых знаний, информации, а в данном случае новых возможностей для пользователей. Т.к. уже существуют аналоги разрабатываемого программного средства, и в то же время программное средство предоставляет новый функционал и новые возможности, данный критерий можно классифицировать на 3-ий уровень качества.

Значимость для науки и практики. Основными чертами этого критерия являются масштабы влияния результатов на сферу применения: от использования на отдельном предприятии до применения в мировом масштабе. Уровень качества значимости для данной НИР - 3, т.к. программное средство можно использовать как в БГУИР, так и в других УВО РБ.

Объективность. Сущностью этого критерия является степень обоснованности результата. Степень объективности может выявляться посредством учета квалификации и компетенции разработчиков и по формам признания результата. Программное средство, разрабатываемое в ходе дипломного проектирования, одобрен деканатом факультета компьютерного проектирования БГУИР, что позволяет классифицировать

объективность на 3-ий уровень.

Доказательность. Степень доказательности результатов может изменяться в пределах неопределенности до возможности воспроизведения и применения на практике. Результат получен на основе теоретических исследований, создания и испытания технологии, изготовленной по рабочей документации процесса. Уровень качества данного критерия - 4.

Точность. Основным содержанием этого критерия является соответствие ПС стандартам, техническому заданию. Созданный экспериментальный образец соответствует техническому заданию и государственному стандарту и не требует доработки. Уровень качества данного критерия - 4.

В таблице 5.3 отображены оценки уровня и значимости критериев качества результата НИР.

Таблица 5.3 - Критерии качества научно-технического результата

|  |  |  |
| --- | --- | --- |
| Наименование критерия | Уровень качества | Значимость |
| 1 | 2 | 3 |
| Новизна | 3 | 0,25 |
| Значимость для науки и практики | 3 | 0,15 |
| Объективность | 3 | 0,15 |
| Доказательность | 4 | 0,15 |
| Точность | 4 | 0,3 |

По данным из таблицы 5.3 можно рассчитать комплексный показатель достигнутого уровня результата выполненных работ:

КК = 3 • 0,25 + 3 • 0,15 + 3 • 0,15 + 4 • 0,15 + 4 • 0,3 = 3,45

Максимально возможное значение комплексного коэффициента качества (достигнутого уровня результата дипломного проекта), взвешенного по всем критериям, равно 5, а в общем случае равно КК ≤ 5).

Чем ближе значение КК ≤ 5, тем выше научно-техническая результативность дипломного проекта. Если КК ≥ 3, то можно считать, что полученные в дипломном проекте результаты соответствуют современным требованиям.

Если КК < 3, то можно считать, что выполненный дипломный проект не

удовлетворяет по уровню качества разработок современным требованиям.

На основе полученной оценки уровня качества данного дипломного проекта (равной 3,45), можно утверждать, что выполненный дипломный проект удовлетворяет по уровню качества разработок современным требованиям.

B результате технико-экономического обоснования разработки программного продукта были получены следующие результаты:

* получена смета затрат на разработку и адаптацию, которая составила 17238,342 руб.;
* рассчитана оценка уровня качества, которая составила 3,45.

Исходя из полученных данных можно сделать вывод, что разработанный проект является эффективным и затраты, связанные с разработкой программного продукта, экономически целесообразны.

# ЗАКЛЮЧЕНИЕ

# 

Было выполнено пред-проектное исследование предметной области и определены основные бизнес-процессы системы визовой поддержки иностранных студентов. На основании одного из бизнес-процессов были получены практические навыки построения и описания IDEF0 и BPMN моделей до и после автоматизации.

По завершению проделанной работы можно сделать вывод: стадия проектирования важна и необходима, так как разработчики получают согласованные четкие планы действий, что сокращает время разработки системы и повышает её качество.

Анализ предметной области, позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта.

Описание существующих бизнес-процессов помогает выявить направления для автоматизации, разработать набор необходимой функциональности будущей системы.

В ходе выполнения лабораторной работы было выполнено построение информационной модели предметной области, разработаны и описаны модели поведения и взаимодействия объектов программного средства, осуществлена постановка задачи на разработку.

По завершению проделанной работы можно сделать вывод: стадия проектирования важна и необходима, так как разработчики получают согласованные четкие планы действий, что сокращает время разработки системы и повышает её качество.

Информационная модель данных предназначена для представления семантики предметной области в терминах субъективных средств описания – сущностей, атрибутов, идентификаторов сущностей, супертипов, подтипов и т. д.

В спецификации требований к программному средству должна быть представлена детализация функций проектируемого программного средства с учетом требований к данным, интерфейсам, атрибутам качества, ограничениям и других требований.

В ходе выполнения работы были построены и описаны диаграммы состояний и последовательности. Диаграмма состояний необходима для моделирования состояния программных объектов системы, а также переходы между этими состояниями. Диаграмма последовательности необходима для моделирования, упорядоченного во времени взаимодействия объектов системы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Дж. Рамбо, А. Джекобсон. – М. : ДМК Пресс, 2004. – 432 с.
2. Онлайн-курс обучения программированию на Java [Электронный ресурс]. – Режим доступа : https://javarush.ru/groups/posts/1973[-abstraktniheklassih-v-java-na-konkretnihkh-primerakh.](https://javarush.ru/groups/posts/1973-abstraktnihe-klassih-v-java-na-konkretnihkh-primerakh)
3. Руководство по TCP/IP для начинающих [Электронный ресурс]. – Режим доступа : [http://www.codenet.ru/webmast/tcpip.](http://www.codenet.ru/webmast/tcpip)
4. Википедия [Электронный ресурс]. –

Режим доступа : [https://ru.wikipedia.org/wiki/PostgreSQL.](https://ru.wikipedia.org/wiki/PostgreSQL)

1. Сложные ИТ-проекты. Автоматизация [Электронный ресурс]. –

Режим доступа: [https://web-creator.ru/articles/postgresql.](https://web-creator.ru/articles/postgresql)

1. Бизнес-логика [Электронный ресурс]. – Режим доступа : [https://bizlogica.ru/.](https://biz-logica.ru/.30)
2. Тестирование программного обеспечения [Электронный ресурс]. – Режим доступа : [http://www.protesting.ru/testing/.](http://www.protesting.ru/testing/)
3. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS / С. Орлов. – 2-е изд. – СПб.: Питер, 2006. – 500c.
4. Гома, Х. UML Проектирование систем реального времени, параллельных и распределенных приложений/ Слинкин, А.– М. Издательство ДМК Пресс, 2016. – 480 с.
5. Модель компетенций глазами пользователя – hr-portal [Электронный ресурс]. – Режим доступа: <https://ht-lab.ru/knowledge/articles/2053/>
6. Модель компетенций – это не сложно – hr-director [Электронный ресурс]. – Режим доступа: <https://www.hr-director.ru/article/66647-model-kompetentsiy-19-m4>.
7. IDEF [Электронный ресурс]. – Режим доступа: <http://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/idef>
8. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. – 368 с.: ил. (Серия «Библиотека программиста»)
9. Модель компетенций – laws.studio [Электронный ресурс]. – Режим доступа: <https://laws.studio/upravlenie-personalom-uch/model-kompetentsiy-20662.html>
10. Postgresql Documentation [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/>
11. GraphQl Foundation [Электронный ресурс]. – Режим доступа: <https://foundation.graphql.org/>
12. Apollo documentationn [Электронный ресурс]. – Режим доступа: <https://www.apollographql.com/docs/>
13. Express API reference [Электронный ресурс]. – Режим доступа: <https://expressjs.com/en/4x/api.html>
14. Uml-диаграммы [Электронный ресурс]. – Режим доступа: <http://bourabai.kz/dbt/uml/ch17.htm>
15. React documentation [Электронный ресурс]. – Режим доступа: <https://reactjs.org/docs/getting-started.html>

# ПРИЛОЖЕНИЕ А (обязательное) Отчет о проверке на заимствования в программном средстве «Антиплагиат»

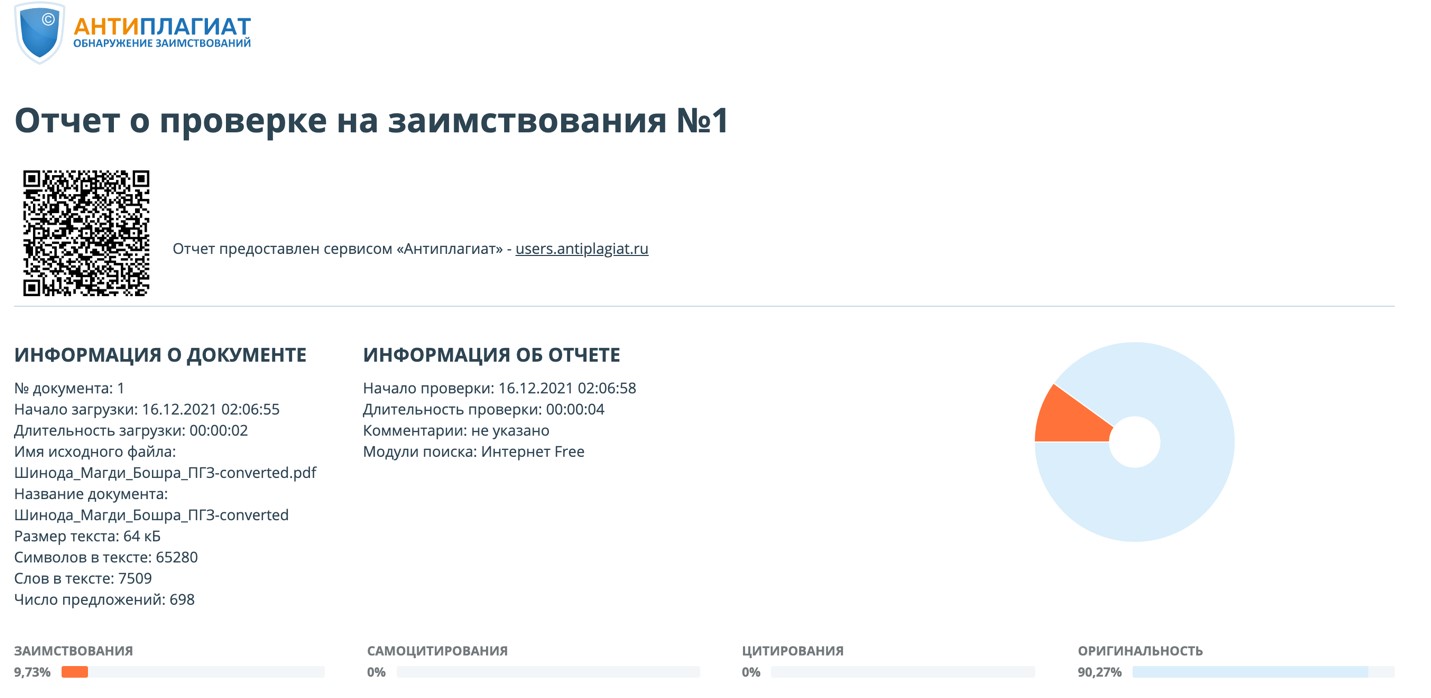


Рисунок А.1 – Отчет о проверке на заимствования в программном средстве «Антиплагиат»

**ПРИЛОЖЕНИЕ Б** **(обязательное)  
Листинг кода алгоритмов, реализующих бизнес-логику**

PACKAGE COM.MAGDY.VISA\_OFFICE.ENTITY;

IMPORT JAVA.UTIL.OBJECTS;

PUBLIC CLASS PERSON { PRIVATE LONG ID; PRIVATE STRING NAME; PRIVATE ROLE ROLE; PRIVATE USER USER; PRIVATE STRING LASTNAME; PRIVATE STRING PHONENUMBER; PRIVATE STRING EMAIL;

PUBLIC PERSON() {

}

PUBLIC PERSON(LONG ID, STRING NAME, ROLE ROLE, USER USER, STRING LASTNAME, STRING PHONENUMBER, STRING EMAIL) { THIS.ID = ID; THIS.NAME = NAME; THIS.ROLE = ROLE; THIS.USER = USER; THIS.LASTNAME = LASTNAME; THIS.PHONENUMBER = PHONENUMBER; THIS.EMAIL = EMAIL;

} PUBLIC PERSON(LONG ID\_PERSON) { THIS.ID = ID\_PERSON;

}

PUBLIC LONG GETID() { RETURN ID;

}

PUBLIC VOID SETID(LONG ID) { THIS.ID = ID;

}

PUBLIC STRING GETNAME() { RETURN NAME;

} PUBLIC VOID SETNAME(STRING NAME) { THIS.NAME = NAME;

} PUBLIC ROLE GETROLE() { RETURN ROLE;

}

PUBLIC VOID SETROLE(ROLE ROLE) { THIS.ROLE = ROLE;

}

PUBLIC USER GETUSER() { RETURN USER;

} PUBLIC VOID SETUSER(USER USER) { THIS.USER = USER;

} PUBLIC STRING GETLASTNAME() { RETURN LASTNAME;

} PUBLIC VOID SETLASTNAME(STRING LASTNAME) { THIS.LASTNAME = LASTNAME;

} PUBLIC STRING GETPHONENUMBER() { RETURN PHONENUMBER;

} PUBLIC VOID SETPHONENUMBER(STRING PHONENUMBER) { THIS.PHONENUMBER = PHONENUMBER;

} PUBLIC STRING GETEMAIL() { RETURN EMAIL;

} PUBLIC VOID SETEMAIL(STRING EMAIL) { THIS.EMAIL = EMAIL;

}

@OVERRIDE PUBLIC BOOLEAN EQUALS(OBJECT O) { IF (THIS == O) RETURN TRUE;

IF (O == NULL || GETCLASS() != O.GETCLASS()) RETURN FALSE; PERSON PERSON = (PERSON) O;

RETURN ID == PERSON.ID && OBJECTS.EQUALS(NAME, PERSON.NAME) && OBJECTS.EQUALS(ROLE, PERSON.ROLE) && OBJECTS.EQUALS(USER, PERSON.USER) && OBJECTS.EQUALS(LASTNAME, PERSON.LASTNAME) && OBJECTS.EQUALS(PHONENUMBER, PERSON.PHONENUMBER) && OBJECTS.EQUALS(EMAIL, PERSON.EMAIL);

}

@OVERRIDE

PUBLIC INT HASHCODE() { RETURN OBJECTS.HASH(ID, NAME, ROLE, USER, LASTNAME, PHONENUMBER, EMAIL);

}

@OVERRIDE PUBLIC STRING TOSTRING() { RETURN "PERSON{" + "ID=" + ID +

", NAME='" + NAME + '\'' +

", ROLE=" + ROLE +

", USER=" + USER +

", LASTNAME='" + LASTNAME + '\'' +

", PHONENUMBER='" + PHONENUMBER + '\'' +

", EMAIL='" + EMAIL + '\'' +

'}';

}

}

PACKAGE COM.MAGDY.VISA\_OFFICE.DAO.IMPL;

IMPORT COM.MAGDY.VISA\_OFFICE.DAO.PERSONDAO; IMPORT COM.MAGDY.VISA\_OFFICE.ENTITY.PERSON; IMPORT COM.MAGDY.VISA\_OFFICE.ENTITY.ROLE; IMPORT COM.MAGDY.VISA\_OFFICE.ENTITY.USER; IMPORT COM.MAGDY.VISA\_OFFICE.EXCEPTION.DAOEXCEPTION;

IMPORT JAVA.SQL.PREPAREDSTATEMENT; IMPORT JAVA.SQL.RESULTSET; IMPORT JAVA.SQL.SQLEXCEPTION; IMPORT JAVA.SQL.STATEMENT; IMPORT JAVA.UTIL.ARRAYLIST; IMPORT JAVA.UTIL.LIST; IMPORT JAVA.UTIL.LOGGING.LOGGER;

PUBLIC CLASS PERSONDAOIMPL EXTENDS BASEDAO IMPLEMENTS PERSONDAO {

PRIVATE STATIC FINAL LOGGER LOGGER =

LOGGER.GETLOGGER(PERSONDAOIMPL.CLASS.GETNAME()); PRIVATE STATIC FINAL STRING INSERT\_PERSON = "INSERT INTO PERSON

(NAME,ROLE\_ID,USER\_ID,LAST\_NAME,PHONE\_NUMBER, EMAIL ) VALUES (?,?,?,?,?,?)";

PRIVATE STATIC FINAL STRING SELECT\_PERSON = "SELECT ID,NAME,ROLE\_ID,USER\_ID,LAST\_NAME,PHONE\_NUMBER, EMAIL FROM PERSON WHERE ID=?"; PRIVATE STATIC FINAL STRING SELECT\_ALL\_PERSON = "SELECT \* FROM PERSON"; PRIVATE STATIC FINAL STRING UPDATE\_PERSON = "UPDATE PERSON SET NAME=?,ROLE\_ID=?,USER\_ID=?,LAST\_NAME=?,PHONE\_NUMBER=?, EMAIL=? WHERE ID=?"; PRIVATE STATIC FINAL STRING DELETE\_PERSON = "DELETE FROM PERSON WHERE ID=?";

@OVERRIDE PUBLIC PERSON SAVE(PERSON PERSON) THROWS DAOEXCEPTION { LOGGER.INFO(" PERSON HAS CREAT ...");

TRY (PREPAREDSTATEMENT PREPAREDSTATEMENT = CONNECTION.PREPARESTATEMENT(INSERT\_PERSON, STATEMENT.RETURN\_GENERATED\_KEYS)) {

PREPAREDSTATEMENT.SETSTRING(1, PERSON.GETNAME()); PREPAREDSTATEMENT.SETLONG(2, PERSON.GETROLE().GETID()); PREPAREDSTATEMENT.SETLONG(3, PERSON.GETUSER().GETID()); PREPAREDSTATEMENT.SETSTRING(4, PERSON.GETLASTNAME()); PREPAREDSTATEMENT.SETSTRING(5, PERSON.GETPHONENUMBER()); PREPAREDSTATEMENT.SETSTRING(6, PERSON.GETEMAIL());

INT REWSAVE = PREPAREDSTATEMENT.EXECUTEUPDATE(); IF (REWSAVE > 0) {

TRY (RESULTSET GENERATEDKEYS = PREPAREDSTATEMENT.GETGENERATEDKEYS()) { IF (GENERATEDKEYS.NEXT()) {

PERSON.SETID(GENERATEDKEYS.GETLONG(1)); RETURN PERSON;

} ELSE { THROW NEW DAOEXCEPTION("CREATING ROLE FAILED..."); }

}

}

} CATCH (SQLEXCEPTION E) {

THROW NEW DAOEXCEPTION(E.GETMESSAGE());

} RETURN NULL;

}

@OVERRIDE PUBLIC PERSON SELECTBYID(LONG ID) THROWS DAOEXCEPTION { LOGGER.INFO("SELECTED BY ID ..."); PERSON PERSON = NULL;

TRY (PREPAREDSTATEMENT PREPAREDSTATEMENT = CONNECTION.PREPARESTATEMENT(SELECT\_PERSON)) { PREPAREDSTATEMENT.SETLONG(1, ID);

RESULTSET RESULTSET = PREPAREDSTATEMENT.EXECUTEQUERY(); WHILE (RESULTSET.NEXT()) {

STRING NAME = RESULTSET.GETSTRING("NAME"); LONG ROLEID = RESULTSET.GETLONG("ROLE\_ID"); LONG USERID = RESULTSET.GETLONG("USER\_ID");

STRING LASTNAME = RESULTSET.GETSTRING("LAST\_NAME");

STRING PHONENUMBER = RESULTSET.GETSTRING("PHONE\_NUMBER");

STRING EMAIL = RESULTSET.GETSTRING("EMAIL");

PERSON = NEW PERSON(ID, NAME, NEW ROLE(ROLEID), NEW

USER(USERID), LASTNAME, PHONENUMBER, EMAIL);

}

} CATCH (SQLEXCEPTION E) { THROW NEW DAOEXCEPTION(E.GETMESSAGE());

} RETURN PERSON;

}

@OVERRIDE

PUBLIC LIST<PERSON> SELECTALL() THROWS DAOEXCEPTION { LOGGER.INFO(" SELECTED ALL PERSON ..."); LIST<PERSON> PERSONLIST = NEW ARRAYLIST<>();

TRY (PREPAREDSTATEMENT PREPAREDSTATEMENT =

CONNECTION.PREPARESTATEMENT(SELECT\_ALL\_PERSON)) {

RESULTSET RESULTSET = PREPAREDSTATEMENT.EXECUTEQUERY(); WHILE (RESULTSET.NEXT()) { LONG ID = RESULTSET.GETLONG("ID"); STRING NAME = RESULTSET.GETSTRING("NAME"); LONG ROLEID = RESULTSET.GETLONG("ROLE\_ID"); LONG USERID = RESULTSET.GETLONG("USER\_ID");

STRING LASTNAME = RESULTSET.GETSTRING("LAST\_NAME");

STRING PHONENUMBER = RESULTSET.GETSTRING("PHONE\_NUMBER");

STRING EMAIL = RESULTSET.GETSTRING("EMAIL");

PERSONLIST.ADD(NEW PERSON(ID, NAME, NEW ROLE(ROLEID), NEW

USER(USERID), LASTNAME, PHONENUMBER, EMAIL));

}

} CATCH (SQLEXCEPTION E) { THROW NEW DAOEXCEPTION(E.GETMESSAGE());

} RETURN PERSONLIST;

}

|  |  |
| --- | --- |
| @OVERRIDE PUBLIC PERSON UPDATE(PERSON PERSON) THROWS DAOEXCEPTION { LOGGER.INFO("PERSON HAS UPDATE ...."); |  |
| TRY (PREPAREDSTATEMENT PREPAREDSTATEMENT  CONNECTION.PREPARESTATEMENT(UPDATE\_PERSON)) {  PREPAREDSTATEMENT.SETSTRING(1, PERSON.GETNAME()); PREPAREDSTATEMENT.SETLONG(2, PERSON.GETROLE().GETID()); PREPAREDSTATEMENT.SETLONG(3, PERSON.GETUSER().GETID()); PREPAREDSTATEMENT.SETSTRING(4, PERSON.GETLASTNAME()); PREPAREDSTATEMENT.SETSTRING(5, PERSON.GETPHONENUMBER()); PREPAREDSTATEMENT.SETSTRING(6, PERSON.GETEMAIL()); PREPAREDSTATEMENT.SETLONG(7, PERSON.GETID()); PREPAREDSTATEMENT.EXECUTEUPDATE(); RETURN PERSON;    } CATCH (SQLEXCEPTION E) { THROW NEW DAOEXCEPTION(E.GETMESSAGE());  }  }    @OVERRIDE PUBLIC BOOLEAN DELETEBYID(LONG ID) THROWS DAOEXCEPTION { LOGGER.INFO(" PERSON HAS DELETE ..."); BOOLEAN REWDELETE = FALSE; | = |
| TRY (PREPAREDSTATEMENT PREPAREDSTATEMENT  CONNECTION.PREPARESTATEMENT(DELETE\_PERSON)) { PREPAREDSTATEMENT.SETLONG(1, ID); REWDELETE = PREPAREDSTATEMENT.EXECUTEUPDATE() > 0; | = |

} CATCH (SQLEXCEPTION E) {

THROW NEW DAOEXCEPTION(E.GETMESSAGE());

} RETURN REWDELETE;

}

}

PACKAGE COM.MAGDY.VISA\_OFFICE.SERVICE.IMPL;

IMPORT COM.MAGDY.VISA\_OFFICE.DAO.IMPL.ROLEDAOIMPL; IMPORT COM.MAGDY.VISA\_OFFICE.DAO.IMPL.USERDAOIMPL; IMPORT COM.MAGDY.VISA\_OFFICE.ENTITY.PERSON; IMPORT COM.MAGDY.VISA\_OFFICE.ENTITY.ROLE;

IMPORT COM.MAGDY.VISA\_OFFICE.ENTITY.USER; IMPORT COM.MAGDY.VISA\_OFFICE.EXCEPTION.DAOEXCEPTION; IMPORT COM.MAGDY.VISA\_OFFICE.EXCEPTION.SERVICEEXCEPTION; IMPORT COM.MAGDY.VISA\_OFFICE.SERVICE.BASESERVICE; IMPORT COM.MAGDY.VISA\_OFFICE.SERVICE.PERSONSERVICE; IMPORT ORG.APACHE.LOGGING.LOG4J.LOGMANAGER;

IMPORT JAVA.RMI.SERVEREXCEPTION; IMPORT JAVA.UTIL.LIST;

PUBLIC CLASS PERSONSERVICEIMPL EXTENDS BASESERVICE IMPLEMENTS PERSONSERVICE {

PRIVATE FINAL ORG.APACHE.LOGGING.LOG4J.LOGGER LOGGER =

LOGMANAGER.GETLOGGER(PERSONSERVICEIMPL.CLASS);

PRIVATE FINAL PERSON PERSON = NEW PERSON(); PRIVATE FINAL PERSONSERVICEIMPL PERSONSERVICE = NEW PERSONSERVICEIMPL(); PRIVATE FINAL

ROLEDAOIMPL ROLEDAO = NEW ROLEDAOIMPL(); PRIVATE FINAL USERDAOIMPL USERDAO = NEW USERDAOIMPL();

@OVERRIDE PUBLIC PERSON SAVE(PERSON PERSON) THROWS SERVICEEXCEPTION { LOGGER.INFO("DIRECTOR SAVED"); TRY { RETURN PERSONSERVICE.SAVE(PERSON); } CATCH (SERVICEEXCEPTION E) { THROW NEW SERVICEEXCEPTION(E.GETMESSAGE());

}

}

@OVERRIDE PUBLIC PERSON SELECTBYID(LONG ID) THROWS SERVICEEXCEPTION { LOGGER.INFO("DIRECTOR SAVED"); TRY {

LONG ROLEID =PERSON.GETROLE().GETID(); LONG USERID =PERSON.GETUSER().GETID();

ROLE ROLE =ROLEDAO.SELECTBYID(ROLEID);

USER USER =USERDAO.SELECTBYID(USERID);

PERSON.SETROLE(ROLE); PERSON.SETUSER(USER);

RETURN PERSON; } CATCH (DAOEXCEPTION E) { THROW NEW SERVICEEXCEPTION(E.GETMESSAGE());

}

}

@OVERRIDE PUBLIC LIST<PERSON> SELECTALL() THROWS SERVICEEXCEPTION { LOGGER.INFO("DIRECTOR SAVED"); TRY { RETURN PERSONSERVICE.SELECTALL(); } CATCH (SERVICEEXCEPTION E) { THROW NEW SERVICEEXCEPTION(E.GETMESSAGE());

}

}

@OVERRIDE PUBLIC PERSON UPDATE(PERSON PERSON) THROWS SERVICEEXCEPTION { LOGGER.INFO("DIRECTOR SAVED"); TRY { RETURN PERSONSERVICE.UPDATE(PERSON);

} CATCH (SERVICEEXCEPTION E) { THROW NEW SERVICEEXCEPTION(E.GETMESSAGE());

}

}

@OVERRIDE PUBLIC BOOLEAN DELETEBYID(LONG ID) THROWS SERVICEEXCEPTION { LOGGER.INFO("DIRECTOR SAVED"); TRY { RETURN PERSONSERVICE.DELETEBYID(ID);

} CATCH (SERVICEEXCEPTION E) { THROW NEW SERVICEEXCEPTION(E.GETMESSAGE());

}

}

}

PACKAGE COM.MAGDY.VISA\_OFFICE.DAO.CONNECTION;

IMPORT JAVA.SQL.CONNECTION; IMPORT JAVA.SQL.DRIVERMANAGER; IMPORT JAVA.SQL.SQLEXCEPTION; IMPORT JAVA.UTIL.LOGGING.LOGGER;

PUBLIC CLASS CONNECTIONBUILDER {

|  |  |  |
| --- | --- | --- |
| PRIVATE STATIC FINAL LOGGER  LOGGER.GETLOGGER(CONNECTIONBUILDER.CLASS.GETNAME()); | LOGGER | = |
| PRIVATE STATIC STRING | JDBCURL | = |

"JDBC:POSTGRESQL://LOCALHOST:5432/VISA?USESSL=FALSE"; PRIVATE STATIC STRING JDBCUSERNAME = "POSTGRES"; PRIVATE STATIC STRING JDBCPASSWORD = "12345"; PUBLIC STATIC CONNECTION GETCONNECTION() { TRY {

CONNECTION CONNECTION = DRIVERMANAGER.GETCONNECTION(JDBCURL, JDBCUSERNAME, JDBCPASSWORD);

LOGGER.INFO("CONNECTED....."); RETURN CONNECTION;

} CATCH (SQLEXCEPTION E) { THROW NEW ERROR("THERE A PROBLEM.....", E);

}

}

}

PACKAGE COM.MAGDY.VISA\_OFFICE.CONTROLLER;

IMPORT COM.MAGDY.VISA\_OFFICE.COMMEND.COMMANDFACTORY; IMPORT COM.MAGDY.VISA\_OFFICE.COMMEND.COMMEND; IMPORT COM.MAGDY.VISA\_OFFICE.COMMEND.PAGE; IMPORT COM.MAGDY.VISA\_OFFICE.EXCEPTION.SERVICEEXCEPTION;

IMPORT JAVAX.SERVLET.REQUESTDISPATCHER; IMPORT JAVAX.SERVLET.SERVLETEXCEPTION; IMPORT JAVAX.SERVLET.HTTP.HTTPSERVLET; IMPORT JAVAX.SERVLET.HTTP.HTTPSERVLETREQUEST; IMPORT JAVAX.SERVLET.HTTP.HTTPSERVLETRESPONSE; IMPORT JAVA.IO.IOEXCEPTION; IMPORT JAVA.UTIL.LOGGING.LOGGER;

PUBLIC CLASS FRONTCONTROLLER EXTENDS HTTPSERVLET {

PRIVATE STATIC FINAL LOGGER LOGGER =

LOGGER.GETLOGGER(FRONTCONTROLLER.CLASS.GETNAME());

PRIVATE STRING MESSAGE;

PUBLIC VOID DOGET(HTTPSERVLETREQUEST REQ, HTTPSERVLETRESPONSE RESP) THROWS SERVLETEXCEPTION { PROCESS(REQ, RESP);

}

@OVERRIDE PROTECTED VOID DOPOST(HTTPSERVLETREQUEST REQUEST, HTTPSERVLETRESPONSE RESPONSE) THROWS SERVLETEXCEPTION { PROCESS(REQUEST, RESPONSE);

} PUBLIC VOID PROCESS(HTTPSERVLETREQUEST REQUEST, HTTPSERVLETRESPONSE RESPONSE) THROWS SERVLETEXCEPTION {

COMMEND COMMAND = COMMANDFACTORY.GETCOMMAND(REQUEST); IF(COMMAND==NULL){ LOGGER.INFO("NO COMMEND HAS BEEN FOIND "); RETURN ;

} TRY { PAGE PAGE = COMMAND.EXECUTE(REQUEST); IF (PAGE.ISREDIRECT()) { REDIRECT(PAGE.GETURL(), REQUEST, RESPONSE);

} ELSE { FORWARD(PAGE.GETURL(), REQUEST, RESPONSE);

}

} CATCH (IOEXCEPTION | SERVICEEXCEPTION E) { THROW NEW SERVLETEXCEPTION(E.GETMESSAGE(), E);

} }

PRIVATE VOID REDIRECT(STRING URL, HTTPSERVLETREQUEST REQUEST, HTTPSERVLETRESPONSE RESPONSE) THROWS IOEXCEPTION { RESPONSE.SENDREDIRECT(REQUEST.GETCONTEXTPATH() + URL);

}

PRIVATE VOID FORWARD(STRING URL, HTTPSERVLETREQUEST REQUEST, HTTPSERVLETRESPONSE RESPONSE) THROWS SERVLETEXCEPTION, IOEXCEPTION {

REQUESTDISPATCHER REQUESTDISPATCHER = REQUEST.GETREQUESTDISPATCHER(URL); REQUESTDISPATCHER.FORWARD(REQUEST, RESPONSE);

}

}

# ПРИЛОЖЕНИЕ В (обязательное) Листинг скрипта генерации базы данных

CREATE TABLE document

(

"id" bigserial NOT NULL, photoNum varchar(50) NOT NULL, order\_universityCode varchar(50) NOT NULL,

CONSTRAINT PK\_5 PRIMARY KEY ("id")

);

CREATE TABLE passport

(

"id" bigserial NOT NULL, name varchar(50) NOT NULL, last\_name varchar(50) NOT NULL, middle\_name varchar(50) NOT NULL, passport\_number varchar(12) NOT NULL, birthday varchar(50) NOT NULL, expiry\_data varchar(50) NOT NULL, country varchar(50) NOT NULL, living\_place varchar(50) NOT NULL,

CONSTRAINT PK\_21 PRIMARY KEY ("id")

);

CREATE TABLE users

(

"id" bigserial NOT NULL, login varchar(50) NOT NULL, password varchar(50) NOT NULL,

CONSTRAINT PK\_136 PRIMARY KEY ("id")

);

CREATE TABLE role

(

"id" bigserial NOT NULL, roleName varchar(50) NOT NULL,

CONSTRAINT PK\_144 PRIMARY KEY ("id")

);

CREATE TABLE person

(

"id" bigserial NOT NULL, name varchar(50) NOT NULL, role\_id bigserial NOT NULL, user\_id bigserial NOT NULL, last\_name varchar(50) NOT NULL, phone\_number varchar(50) NOT NULL, email varchar(50) NOT NULL,

CONSTRAINT PK\_117 PRIMARY KEY ("id"),

CONSTRAINT FK\_139 FOREIGN KEY (user\_id) REFERENCES users ("id"),

CONSTRAINT FK\_146 FOREIGN KEY (role\_id) REFERENCES role ("id")

);

CREATE INDEX FK\_141 ON person

( user\_id );

CREATE INDEX FK\_148 ON person

( role\_id

);

CREATE TABLE director

( id\_person bigserial NOT NULL, "id" bigserial NOT NULL,

CONSTRAINT PK\_10 PRIMARY KEY ("id"),

CONSTRAINT FK\_121 FOREIGN KEY (id\_person) REFERENCES person ("id")

);

CREATE INDEX fkIdx\_123 ON director

( id\_person

);

CREATE TABLE investigator

( id\_person bigserial NOT NULL, "id" bigserial NOT NULL,

CONSTRAINT PK\_54 PRIMARY KEY ("id"),

CONSTRAINT FK\_127 FOREIGN KEY (id\_person) REFERENCES person ("id") );

CREATE INDEX fkIdx\_129 ON investigator

( id\_person

);

CREATE TABLE security

( id\_person bigserial NOT NULL, "id" bigserial NOT NULL,

CONSTRAINT PK\_44 PRIMARY KEY ("id"),

CONSTRAINT FK\_130 FOREIGN KEY (id\_person) REFERENCES person ("id") );

CREATE INDEX fkIdx\_132 ON security

( id\_person

);

CREATE TABLE office

( id\_director bigserial NOT NULL, id\_investigator bigserial NOT NULL, id\_secretary bigserial NOT NULL, "id" bigserial NOT NULL,

CONSTRAINT PK\_97 PRIMARY KEY ("id"),

CONSTRAINT FK\_98 FOREIGN KEY (id\_director) REFERENCES director ("id"),

CONSTRAINT FK\_101 FOREIGN KEY (id\_investigator) REFERENCES investigator ("id"),

CONSTRAINT FK\_104 FOREIGN KEY (id\_secretary) REFERENCES security ("id")

);

CREATE INDEX fkIdx\_100 ON office

(

id\_director

);

CREATE INDEX fkIdx\_103 ON office

(

id\_investigator

);

CREATE INDEX fkIdx\_106 ON office

(

id\_secretary

);

CREATE TABLE student

(

"id" bigserial NOT NULL, id\_passport bigserial NOT NULL, id\_user bigserial NOT NULL, id\_document bigserial NOT NULL,

CONSTRAINT PK\_18 PRIMARY KEY ("id"),

CONSTRAINT FK\_150 FOREIGN KEY (id\_user) REFERENCES users ("id"),

CONSTRAINT FK\_59 FOREIGN KEY (id\_passport) REFERENCES passport ("id"), CONSTRAINT FK\_62 FOREIGN KEY (id\_document) REFERENCES document ("id") );

CREATE INDEX FK\_152 ON student

( id\_user

);

CREATE INDEX fkIdx\_61 ON student

(

id\_passport

);

CREATE INDEX fkIdx\_64 ON student

(

id\_document

);

CREATE TABLE order\_visa

(

id\_student bigserial NOT NULL, id\_office bigserial NOT NULL, "id" bigserial NOT NULL,

CONSTRAINT PK\_41 PRIMARY KEY ("id"),

CONSTRAINT FK\_107 FOREIGN KEY (id\_office) REFERENCES office ("id"), CONSTRAINT FK\_82 FOREIGN KEY (id\_student) REFERENCES student ("id")

);

CREATE INDEX fkIdx\_109 ON order\_visa

( id\_office

);

CREATE INDEX fkIdx\_84 ON order\_visa

( id\_student

);

CREATE TABLE data\_visa

(

"id" bigserial NOT NULL, id\_order bigserial NOT NULL, data\_order varchar(50) NOT NULL, data\_take date NOT NULL, status\_visa boolean NOT NULL,

CONSTRAINT PK\_38 PRIMARY KEY ("id"),

CONSTRAINT FK\_88 FOREIGN KEY (id\_order) REFERENCES order\_visa ("id") );

CREATE INDEX fkIdx\_90 ON data\_visa

( id\_order

);

CREATE TABLE visa

(

"id" bigserial NOT NULL, id\_visa bigserial NOT NULL,

CONSTRAINT PK\_34 PRIMARY KEY ("id"),

CONSTRAINT FK\_92 FOREIGN KEY (id\_visa) REFERENCES data\_visa ("id")

);

CREATE INDEX fkIdx\_94 ON visa

( id\_visa

);

# ВЕДОМОСТЬ КУРСОВОЙ РАБОТЫ