МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

База данных оператора сотовой связи с использованием технологии Advanced Analytics

Выполнил студент Бельский А.А.

(Ф.И.О.)

Руководитель проекта ст. п. Блинова Е. А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Реферат

Лит.

Листов

1

БГТУ

Реферат

Пояснительная записка курсового проекта содержит X страниц пояснительной записки, X источников литературы, 3 приложения.

.NET Framework, WIndows PRESENTATION FOUNDATION, ORACLE SQL SERVER, ADO.NET, Dapper.

Цель курсового проекта – разработка базы данных и приложения, которые обеспечат простое и удобное взаимодействие между собой, что позволит пользователю управлять своими договорами с оператором сотовой связи по предоставляемым им услугам.

В первой главе проводится аналитический обзор схожих приложений по тематике курсового проекта.

Вторая глава посвящена процессу проектирования системы и описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования проекта.

В четвёртой главе описано тестирование, проверка работоспособности и анализ данных системы.

В заключении приведены результаты проделанной работы.

Содержание

[Реферат 2](#_Toc27830404)

[Введение 4](#_Toc27830405)

[1 Аналитический обзор схожих приложений](#_Toc27830406) 5

[2 Проектирование и разработка базы данных](#_Toc27830407) 8

[2.1 Спецификация функциональных требований](#_Toc27830408) 8

[2.2 Разработка модели базы данных](#_Toc27830409) 10

[2.3 Пользователи](#_Toc27830410) 12

[2.4 Хранимые процедуры](#_Toc27830408) 12

[2.5 Реализация технологий](#_Toc27830408) 18

[2.5.1 Реализация технологии мультимедийные типы данных в БД](#_Toc27830409) 18

[3 Проектирование и разработка программного средства](#_Toc27830436) 23

[4 Тестирование, проверка работоспособности и анализ данных 2](#_Toc27830404)5

[Заключение 34](#_Toc27830405)

[Список литературы](#_Toc27830406) 35

[Приложение А](#_Toc27830407) 36

[Приложение Б 37](#_Toc27830405)

[Приложение В](#_Toc27830406) 39

**Введение**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Введение

Лит.

Листов

1

БГТУ

В данной записке приведено описание реляционной базы данных и desktop-приложения, разработанных в соответствии с заданием на курсовое проектирование по теме «База данных оператора сотовой связи с использованием технологии Advanced Analytics».

Задачей данной курсовой работы является разработка базы данных, предназначенной для ведения учёта сотрудников, клиентов, их договоров с сопутствующей информацией, а также для приложения «Менеджер услуг сотовой связи». Приложение будет обладать следующими функционалом:

* авторизация пользователей;
* разделение пользователя на сотрудника и клиента;
* просмотр, удаление и добавление договоров клиентом;
* просмотр, удаление и добавление услуг клиентом;
* просмотр номеров телефона, привязанных к договорам клиентом;
* просмотр и добавление рекомендованных тарифов и услуг клиентом;
* просмотр информации об аккаунте клиентом;
* просмотр клиентов и договоров, которые с ними были подписаны, авторизованным сотрудником.

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Для разработки и управления базой данных курсового проекта использовалась система управления реляционными базами данных Oracle SQL Server, в связи с требованием курсовой работы.

## **1 Обзор технических решений и литературных источников**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

1 Обзор технических решений и литературных источников

Лит.

Листов

1

БГТУ

Для составления технических требований к проекту был проведен анализ похожих программных средств.

Приложение «Мой A1».

Мой А1 – приложение для абонентов сети А1 для управления своим аккаунтом, где бы не находились в любой момент времени.

Приложение «Мой А1» позволяет получать актуальные сообщения о предложениях и акциях компании А1, информацию о своем балансе и остатках минут, SMS, MMS и интернет-трафика, подключать и отключать услуги, изменять свой тарифный план и др.

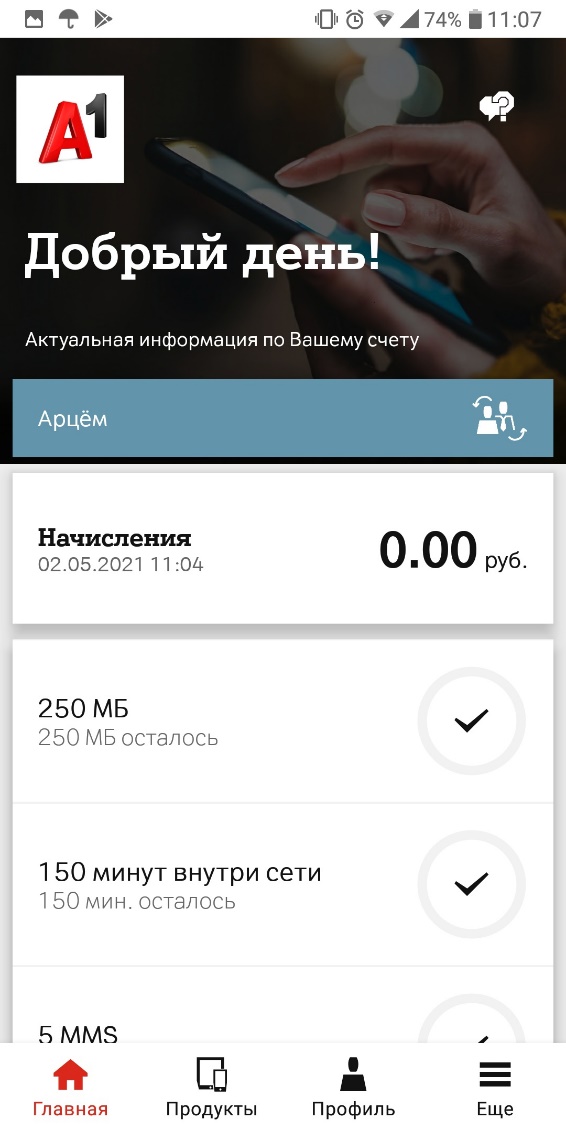


Рисунок 1.1 – Окно приложения «Мой А1»

Приложение «Мой МТС».

Приложение «Мой МТС» — это инструмент самообслуживания абонента, позволяющий получать актуальные данные о состоянии лицевого счета и остатках предоплаченных минут, SMS, MMS и интернет-трафика, подключать/отключать услуги, производить смену тарифного плана и производить другие действия в рамках своего абонентского номера.

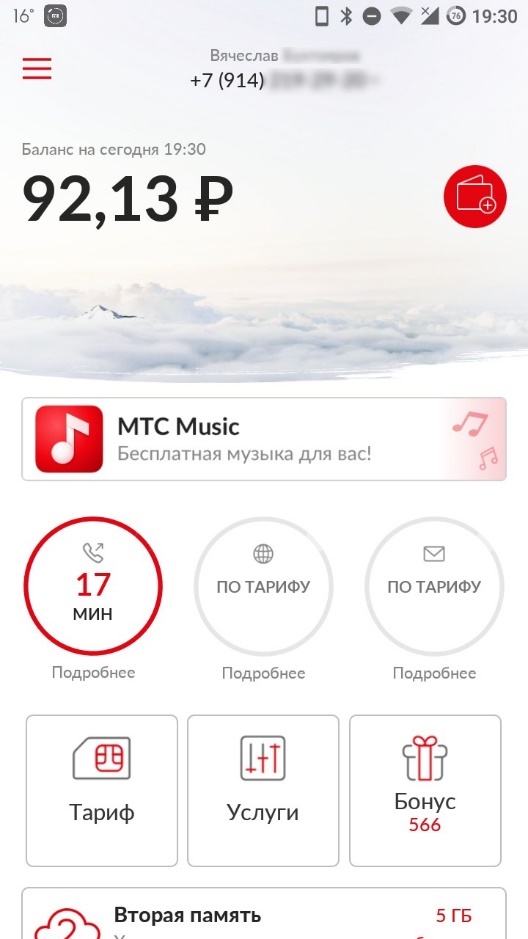


Рисунок 1.2 – Окно приложения «Мой МТС»

Все вышеперечисленные приложения обладают схожим функционалом для управления тарифным планом, услугами и контролем баланса.

**2 Проектирование и разработка базы данных**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

2 Проектирование и разработка базы данных

Лит.

Листов

1

БГТУ

При разработке курсового проекта понадобились следующие объекты:

1. Таблицы;
2. Процедуры, хранимые в пакетах;
3. Индексы;
4. Планировщики с задачами;
5. Триггеры;
6. Представления.

**2.1 Спецификация функциональных требований**

Рассмотрим возможности, которые предоставляет разработанное программное средство. Для наглядности демонстрации использовались UML -диаграммы.

UML – уникальный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

Неавторизованный пользователь ничего не сможет сделать в приложении, только авторизоваться.

Если клиент-пользователь авторизуется, то будет иметь возможности, представленные на рисунке 2.1.1.

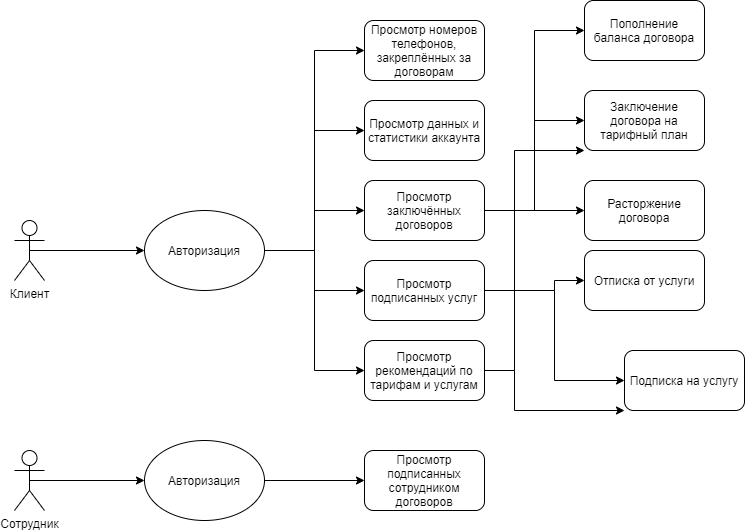


Рисунок 2.1.1 – UML диаграмма вариантов использования для авторизованного клиента

Также в приложение можно войти как сотрудник. Тогда он сможет только просмотреть подписанные с клиентами договора.

Диаграмма вариантов использования для сотрудника отображена на рисунке 2.1.2.

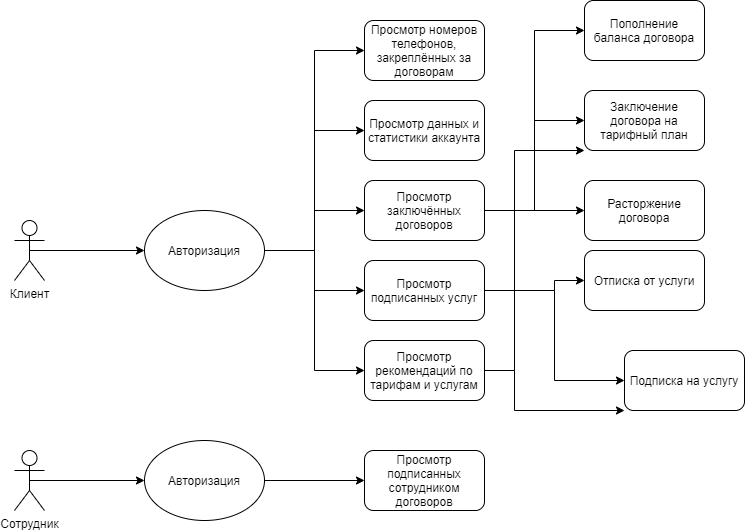


Рисунок 2.1.2 – UML диаграмма вариантов использования для сотрудника

После рассмотрения функциональных требований стоит рассмотреть модель нашей базы данных.

**2.2 Разработка модели базы данных**

Для реализации базы данных для оператора сотовой связи было разработано 13 таблиц. Диаграмма базы со структурой связей представлена в приложении А, а в приложении Б представлены скрипты создания БД.

Таблица Account предназначена для хранения информации о зарегистрированных пользователях, состоит из 3 столбцов:

* login – логин аккаунта, уникальный для каждого пользователя, тип nvrachar2(50), первичный ключ;
* hash\_password – хэш пароля пользователя, захэшированного алгоритмом MD5, тип nvarchar2(32);
* access\_level – уровень доступа пользователя в приложении, тип int.

Таблица Account\_event предназначена для хранения информации о событиях, связанных с аккаунтами (добавление, вход, обновление, удаление), состоит из 4 столбцов:

* id – идентификатор события, тип int, первичный ключ;
* account\_login – логин аккаунта, тип nvarchar2(50);
* event\_datetime – время события, тип timestamp;
* message – сообщение, сопровождающее событие, тип nvarchar2(200).

Таблица Post предназначена для хранения информации о должностях сотрудников, состоит из 3 столбцов:

* id – идентификатор должности, тип int, первичный ключ;
* post\_name – должность, тип nvarchar2(50);
* category – категория должности, тип nvarchar2(50).

Таблица Client предназначена для хранения информации о клиентах, состоит из 4 столбцов:

* id – идентификатор клиента, тип int, первичный ключ;
* full\_name – полное имя клиента, тип nvarchar2(50);
* passport\_number – номер паспорта клиента, тип nvarchar2(12);
* account\_login – логин аккаунта, тип nvarchar2(50), внешний ключ.

Таблица Service\_description предназначена для хранения информации об описании услуг, состоит из 3 столбцов:

* id – идентификатор описания услуги, тип int, первичный ключ;
* service\_name – имя услуги, тип nvarchar2(50);
* service\_description – описание услуги, тип nvarchar2(500).

Таблица Tariff\_plan предназначена для хранения информации об тарифных планах, состоит из 3 столбцов:

* id – идентификатор тарифного плана, тип int, первичный ключ;
* tariff\_name – имя тарифа, тип nvarchar2(50);
* tariff\_amount – стоимость тарифа, тип float(126).

Таблица Employee предназначена для хранения информации о сотрудниках, состоит из 4 столбцов:

* id – идентификатор сотрудника, тип int, первичный ключ;
* full\_name – полное имя сотрудника, тип nvarchar2(50);
* post\_id – идентификатор должности сотрудника, тип int, внешний ключ;
* account\_login – логин аккаунта, тип nvarchar2(50), внешний ключ.

Таблица Contract предназначена для хранения информации о договорах, состоит из 5 столбцов:

* id – идентификатор сотрудника, тип int, первичный ключ;
* tariff\_id – идентификатор тарифа, тип int, внешний ключ;
* client\_id – идентификатор клиента, тип int, внешний ключ;
* employee\_id – идентификатор сотрудника, тип int, внешний ключ;
* signing\_datetime – время подписания, тип timestamp.

Таблица Phone\_number предназначена для хранения информации о номерах телефонов, состоит из 3 столбцов:

* id – идентификатор номера телефона, тип int, первичный ключ;
* phone\_number – номер телефона, тип nvarchar2(14), соответствует регулярному выражению мобильного телефона;
* contract\_id – идентификатор контракта, тип int, внешний ключ.

Таблица Call предназначена для хранения информации о звонках, состоит из 5 столбцов:

* id – идентификатор звонка, тип int, первичный ключ;
* contract\_id – идентификатор контракта, тип int, внешний ключ;
* to\_phone\_number – номер телефона, к которому происходит звонок, тип nvarchar2(14), соответствует регулярному выражению мобильного телефона;
* talk\_time – продолжительность разговора, тип interval day to second;
* call\_datetime – время звонка, тип timestamp.

Таблица Payment предназначена для хранения информации о платежах, состоит из 4 столбцов:

* id – идентификатор оплаты, тип int, первичный ключ;
* contract\_id – идентификатор контракта, тип int, внешний ключ;
* payment\_amount – размер оплаты, тип float(126);
* payment\_datetime – время оплаты, тип timestamp.

Таблица Debit предназначена для хранения информации о списаниях, состоит из 5 столбцов:

* id – идентификатор оплаты, тип int, первичный ключ;
* contract\_id – идентификатор контракта, тип int, внешний ключ;
* debit\_amount – размер списания, тип float(126);
* debit\_datetime – время оплаты, тип timestamp;
* reason – время оплаты, тип nvarchar2(250).

Таблица Service предназначена для хранения информации о существующих сервисах, состоит из 6 столбцов:

* id – идентификатор оплаты, тип int, первичный ключ;
* contract\_id – идентификатор контракта, тип int, внешний ключ;
* description\_id – идентификатор описания, тип int, внешний ключ;
* service\_amount – стоимость сервиса, тип float(126);
* connection\_datetime – время подключения, тип timestamp;
* disconnection\_datetime – время отключения, тип timestamp.

**2.3 Хранимые процедуры**

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

При разработке курсового проекта были созданы процедуры для следующих целей:

1. Добавление строки (для каждой таблицы);
2. Изменение строки (для каждой таблицы);
3. Удаление строки (для каждой таблицы);
4. Получение строки (для каждой таблицы, кроме таблиц аккаунтов);
5. Получение всех строк (для каждой таблицы, кроме таблиц аккаунтов);
6. Получение уровня доступа по логину и паролю аккаунта;
7. Получение клиента по логину аккаунта;
8. Получение сотрудника по логину аккаунта;
9. Получение услуг клиента;
10. Получение договоров клиента;
11. Получение договоров, которые подписал сотрудник;
12. Получение рекомендаций тарифов клиента;
13. Получение баланса контракта;
14. Получение услуг контракта;
15. Получение номеров телефонов, прикреплённых за контрактом;
16. Получение рекомендаций услуг по контракту;
17. Определение и добавление ежемесячных списаний (для планировщика задач);
18. Получение строки номера телефона по полю номера телефона;
19. Удаление просроченных услуг (для планировщика задач).

**2.4 Индексы**

Индекс – объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени.

При разработке курсового проекта были созданы следующие индексы:

1. Индекс по идентификатору контракта и по причине для таблицы Debit (нужен для быстрого нахождения и вставки списаний);
2. Индекс по идентификатору контракта для таблицы Service (нужен для быстрого нахождения сервисов для вставки списаний);
3. Индекс для нахождения строк номеров телефонов по полю номера телефона;
4. Индекс для поиска услуги по дате отключения (для нахождения просроченных услуг).

**2.4 Планировщики задач**

СУБД Oracle – большой и сложный механизм, требующий выполнения определенных плановых работ, таких как сбор статистики о хранимых объектах или сбор/чистка внутренней информации. Необходимость осуществлять плановый запуск работ могут испытывать и пользователи БД.

При разработке курсового проекта были созданы следующие планировщики задач:

1. Планировщик определения и добавления списаний, на основании прошедшего цикла в виде месяца;
2. Планировщик удаления просроченных услуг.

**2.4 Триггеры**

Триггер – хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определённом столбце заданной таблицы реляционной базы данных.

При разработке курсового проекта были созданы следующие триггеры:

1. Добавление в таблицу событий аккаунтов после добавления нового аккаунта;
2. Добавление в таблицу событий аккаунтов после изменения аккаунта;
3. Добавление в таблицу событий аккаунтов после удаления аккаунта;

**2.4 Представления**

Представление – виртуальная таблица, представляющая собой поименованный запрос, который будет подставлен как подзапрос при использовании представления. В отличие от обычных таблиц реляционных баз данных, представление не является самостоятельной частью набора данных, хранящегося в базе.

При разработке курсового проекта были созданы следующие представления:

1. Объединённые поля идентификатора контракта, идентификатора клиента и идентификатора описания сервиса из таблиц Contract и Service (для установки ассоциации в Data Miner);
2. Объединённые поля идентификатора клиента и идентификатора тарифа из таблиц Contract и Tariff\_plan (для установки ассоциации в Data Miner).

**2.5 Реализация технологии**

В процессе разработки проекта была использована технология Advanced Analytics, а точнее её компонента Oracle Data Mining.

Advanced Analytics - дополнительная надстройка к Oracle Database, способная превратить базу данных в полномасштабную аналитическую платформу за счет двух основных компонентов: Oracle Data Mining и Oracle R Enterprise. Опция Oracle Advanced Analytics дает возможность создавать аналитические приложения, а также приложения для стратегического планирования в режиме реального времени. Данный функционал наиболее актуален для разработки аналитики в таких сферах, как динамика прибытия или убытия клиентов, выдвижение рекомендации и предотвращение мошенничества.

Компонента Oracle Data Mining была использована для установки ассоциаций между клиентами, их наборами тарифов, и тарифами, которые скорее всего им также будут нужны, а также ассоциаций между договорами, их услугами, и теми услугами, которые скорее всего подойдут под набор. То есть компонента Oracle Data Mining используется для подбора рекомендаций по тарифным планам и услугам.

Для этого, как было описано выше, были созданы представления (Рисунок 2.5.1).

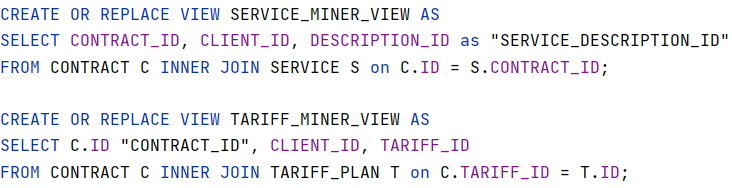


Рисунок 2.5.1 – Представления для установления ассоциаций

После в «Рабочем процессе» (Workflow) добавляем созданные представления, а также компоненты «Ассоциация» (Association), по одной на представление и попарно объединяем (Рисунок 2.5.2).

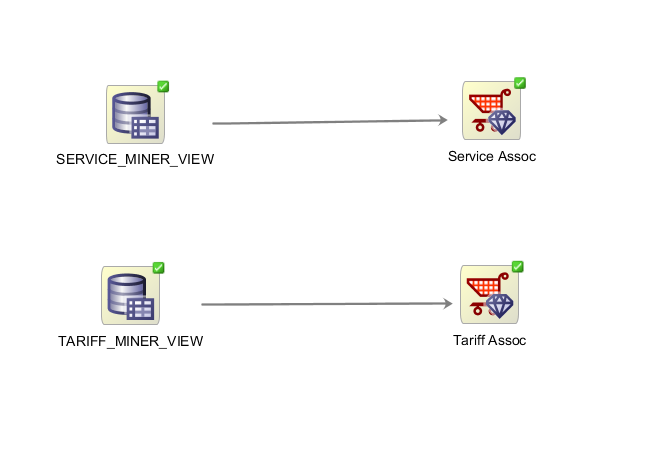


Рисунок 2.5.2 – Связь представлений и компонент ассоциации

Далее в редактировании ассоциации устанавливаем в идентификаторы транзакций (Transaction IDs) элементы представления, с которыми связаны элементы, по которым будут установлены правила ассоциации, затем в идентификаторы предметов (Item ID) сами связанные элементы (Рисунок 2.5.3).

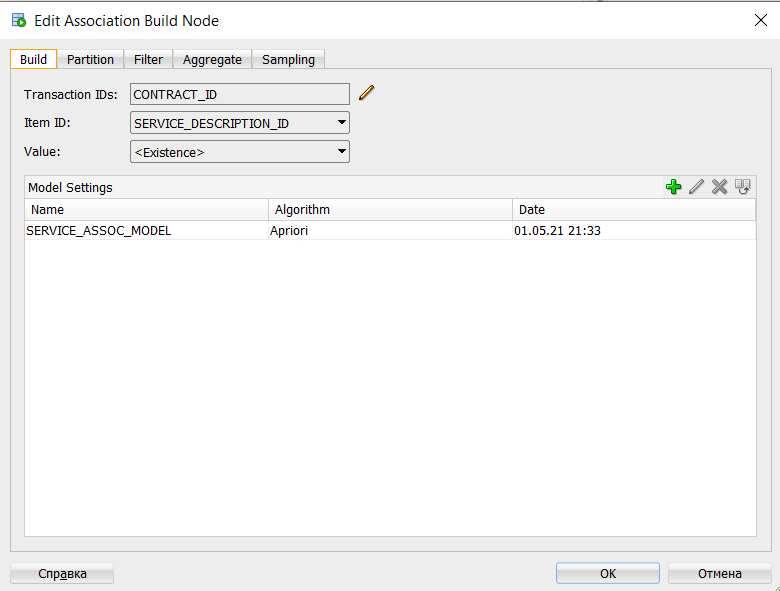


Рисунок 2.5.3 – Окно редактирования ассоциации

После в расширенных настройках (Advanced Settings) устанавливаем параметры максимального размера правила (Maximum rule length), минимальное доверие (Minimum confidence), минимальную поддержку (Minimum support), минимальное размер поддержки (Minimum support count) и минимальную обратное доверие (Minimum reverse confidence) (Рисунок 2.5.4).

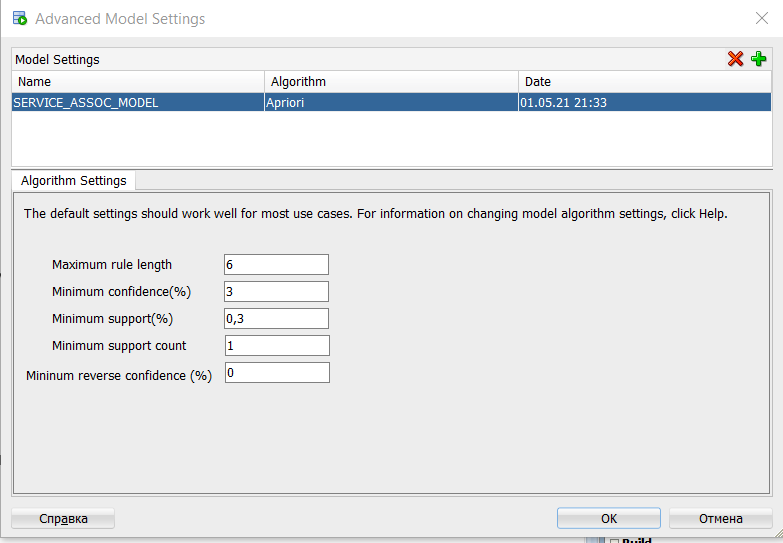


Рисунок 2.5.4 – Окно расширенных настроек ассоциации

Далее запускаем анализ данных (Run) (Рисунок 2.5.5).

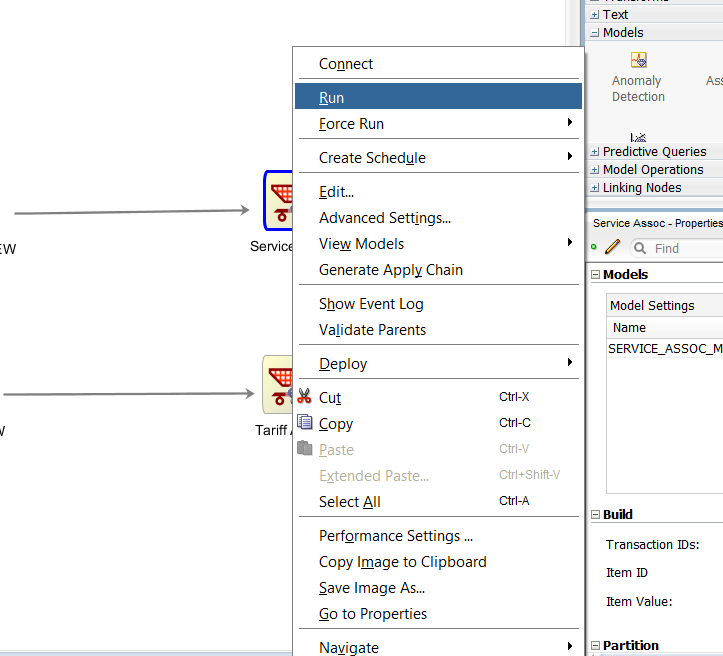


Рисунок 2.5.5 – Контекстное меню с запуском анализа

В результате мы получаем список правил и их характеристик. Для одного набора элементов может быть составлено сразу множество правил, потому лучше отбирать те, у которых показатель Lift выше всего (Рисунок 2.5.6).

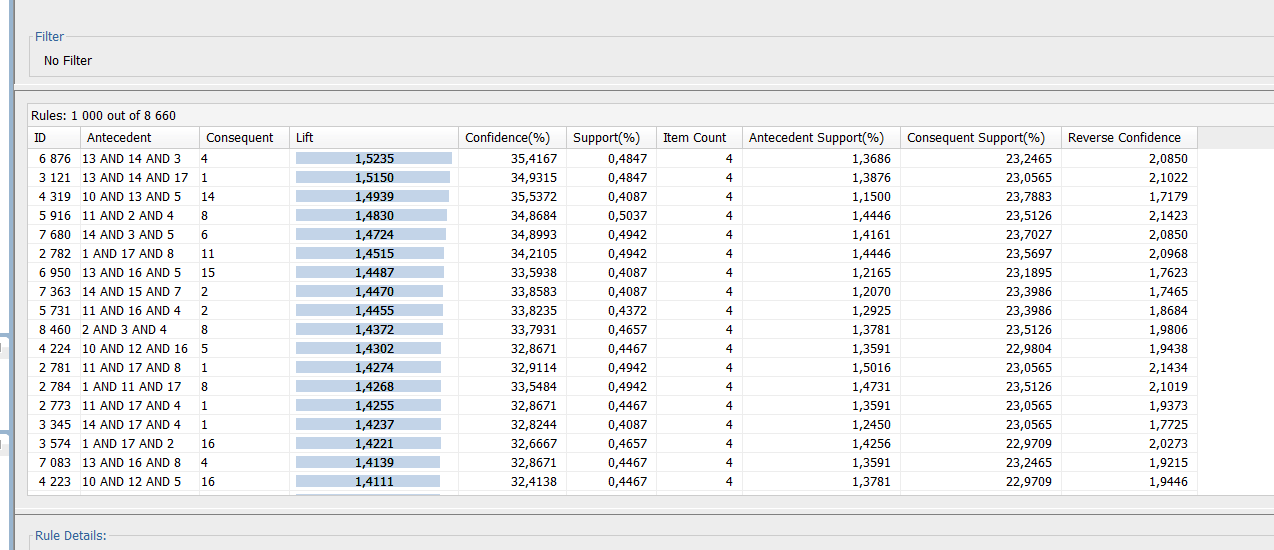


Рисунок 2.5.6 – Список правил ассоциации

В наборе таблиц и представлений появились таблицы и представления со специфическими именами (Рисунок 2.5.7).

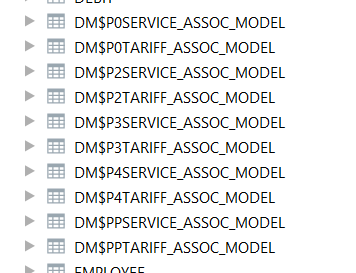


Рисунок 2.5.7 – Сгенерированные таблицы

Именно из них можно извлекать данные для составления рекомендаций.

**2.6 Экспорт и импорт данных**

Для экспорта и импорта данных будет использоваться формат XML. Для этого нужно создать директорию, где будут храниться наши данные (Рисунок 2.6.1).



Рисунок 2.6.1 – Код создания директории

В списке объектов директорий появится объект (Рисунок 2.6.2).

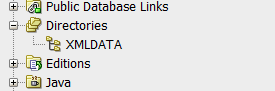


Рисунок 2.6.2 – Объект директории

Используя следующий шаблон, можно экспортировать данные из таблицы в формат XML (Рисунок 2.6.3).

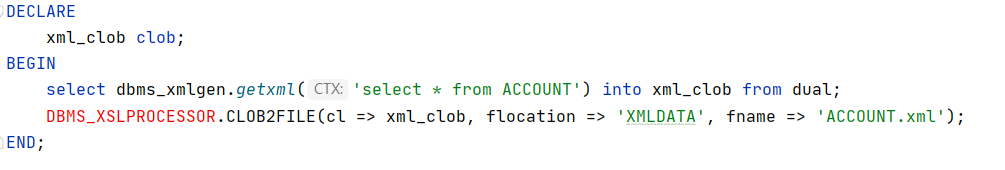


Рисунок 2.6.3 – Шаблон кода для экспортирования данных

Для импортирования данных подойдёт такой шаблон (Рисунок 2.6.4). Этот шаблон состоит из двух частей. В первой части происходит сама вставка данных. Во второй части происходит перемещение последовательности, генерирующей новый числовой идентификатор, в на шаг самого большого существующего в таблице идентификатора.

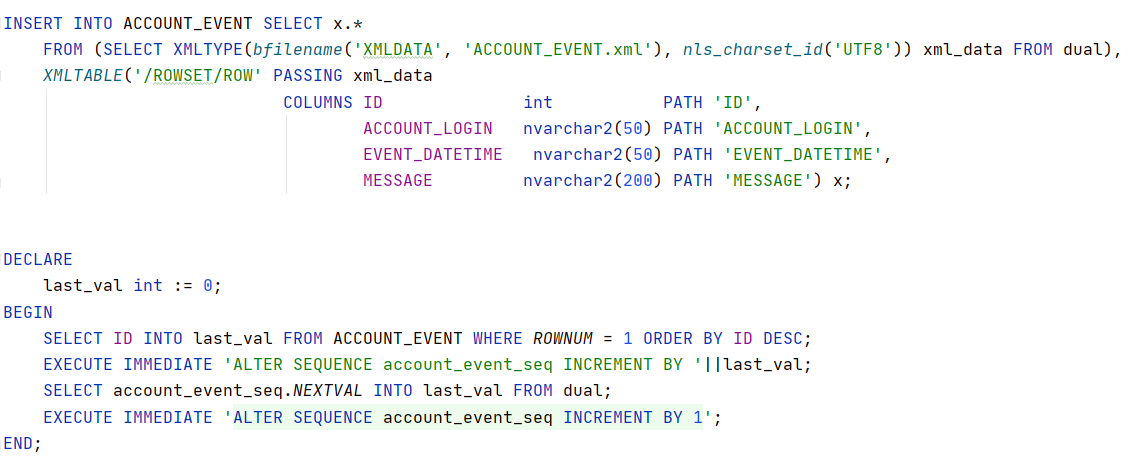


Рисунок 2.6.4 – Шаблон кода для импортирования данных

Отметим, что вставка данных должна проводится в заново созданных таблицах, желательно с обнулёнными последовательностями.

**3 Проектирование и разработка программного средства**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

3 Проектирование и разработка программного средства

Лит.

Листов

1

БГТУ

В процессе выполнения курсового проекта было разработано desktop-приложение, предназначенное для взаимодействия пользователя с базой данных. Приложение было разработано на языке программирования C# с использованием таких технологий как WIndows PRESENTATION FOUNDATION, ADO.NET и ORM Dapper. Структура разработанного программного средства представлена ниже.

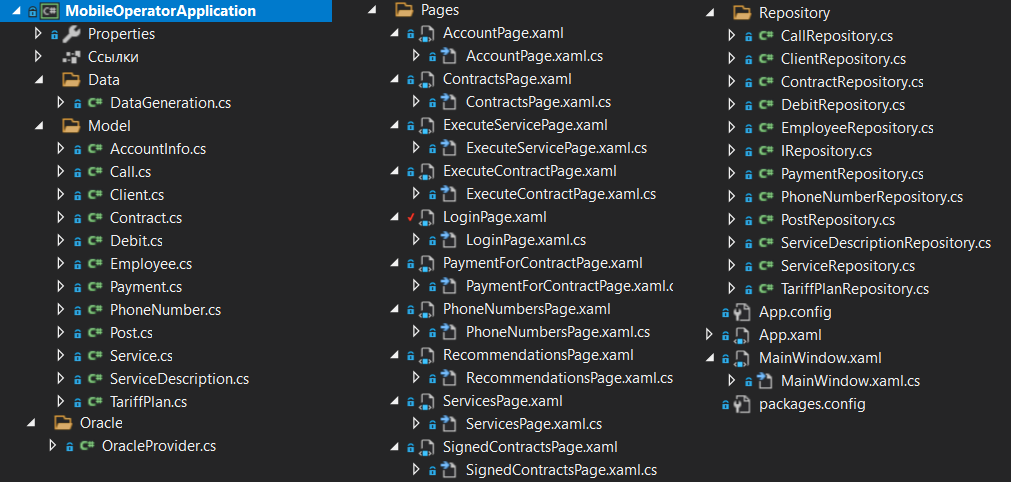


Рисунок 3.1 – Структура проекта

Проект включает в себя следующее окно:

* MainWindow – главное окно c пространством кнопок переключения;

Проект включает в себя следующие страницы:

* AccountPage – страница с информацией об аккаунте клиента;
* ContractsPage – страница со списком договоров клиента;
* ExecuteServicePage – страница подписания услуги для клиента;
* ExecuteContractPage – страница подписания договора для клиента;
* LoginPage – страница входа в аккаунт клиента или сотрудника;
* PaymentForContractPage – страница пополнения баланса контракта;
* PhoneNumbersPage – страница со списком номеров телефона клиента;
* RecommendationsPage – страница рекомендаций для клиента;
* ServicesPage – страница со списком услуг клиента;
* SignedContractsPage – страница подписанных сотрудником договоров.

Для взаимодействия через приложения с данными из БД была применена технология ADO.NET и использована ORM Dapper. ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций. Dapper представляет собой библиотеку с методами расширения для ADO.NET. Схематично архитектуру ADO.NET можно представить следующим образом (Рисунок 3.1).



Рисунок 3.1 – Архитектура ADO.NET

Для установки соединения с нашей базой данных указывается строка подключения в момент создания OracleConnection в OracleProvider (Рисунок 3.2).

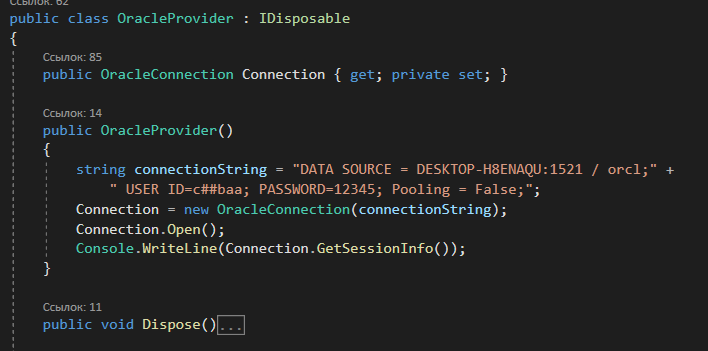


Рисунок 3.2 – Строка подключения

Элементы папки Model предназначены для хранения строк из базы данных. Пример на рисунке 3.3.



Рисунок 3.3 – Пример модели таблицы

Repository содержит в себе классы для управления таблицами, реализующие интерфейс IRepository (Рисунок 3.4). Пример на рисунке 3.5.

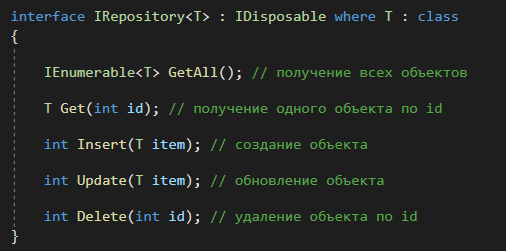


Рисунок 3.4 – Интерфейс IRepository

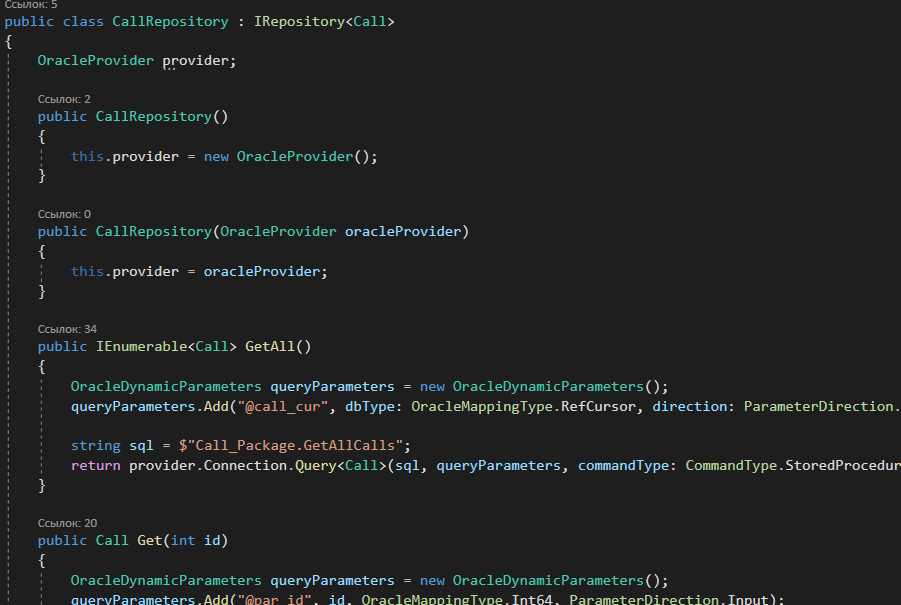


Рисунок 3.5 – Пример менеджера таблицы

Класс DataGeneration предназначен для генерации всей базы данных (Рисунок 3.6).

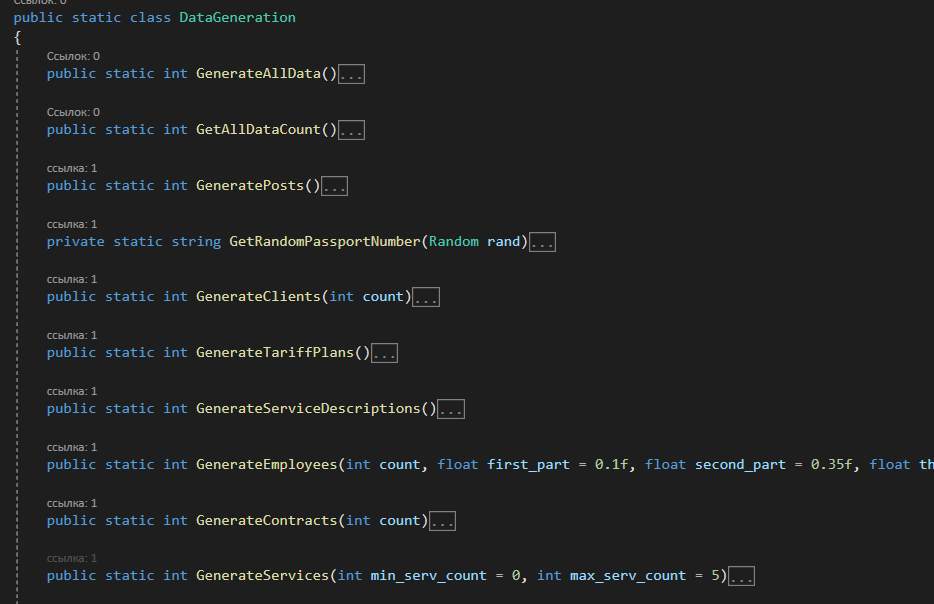


Рисунок 3.6 – Пример менеджера таблицы

Благодаря этому классу и планировщику задач с процедурой по выявлению и добавлению, не совершённых списаний, было сгенерировано более 750 000 строк данных.

**4 Тестирование, проверка работоспособности и анализ данных**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

,Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

4 Тестирование, проверка работоспособности и анализ данных

Лит.

Листов

1

БГТУ

Разработанное программное средство полностью соответствует созданному для него техническому заданию, также оно использует дополнительно разработанный функционал.

Для проверки корректности работы программного средства, продемонстрируем добавление нового пользователя.

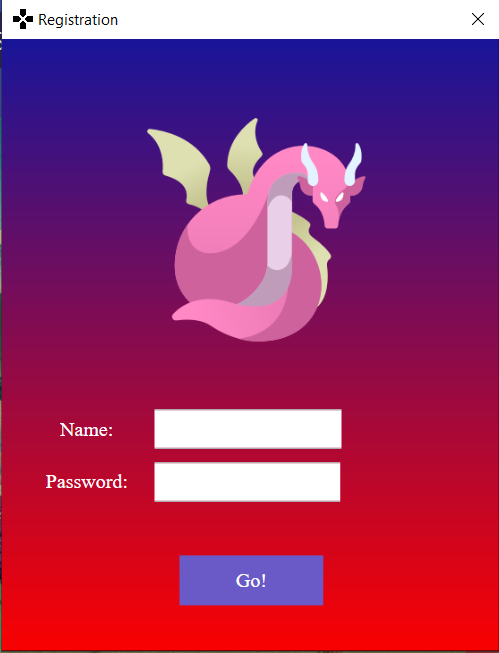


Рисунок 4.1 – Окно регистрации нового пользователя

После нажатия на кнопку «Go!» клиент заносится в базу данных, при этом пароль хэшируется, так как в программе имеется функция хэширования.

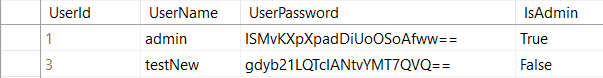


Рисунок 4.2 – Запись только что зарегистрированного пользователя

После того, как пользователь зарегистрировался, открывается начальная страница, показанная на рисунке 4.3.

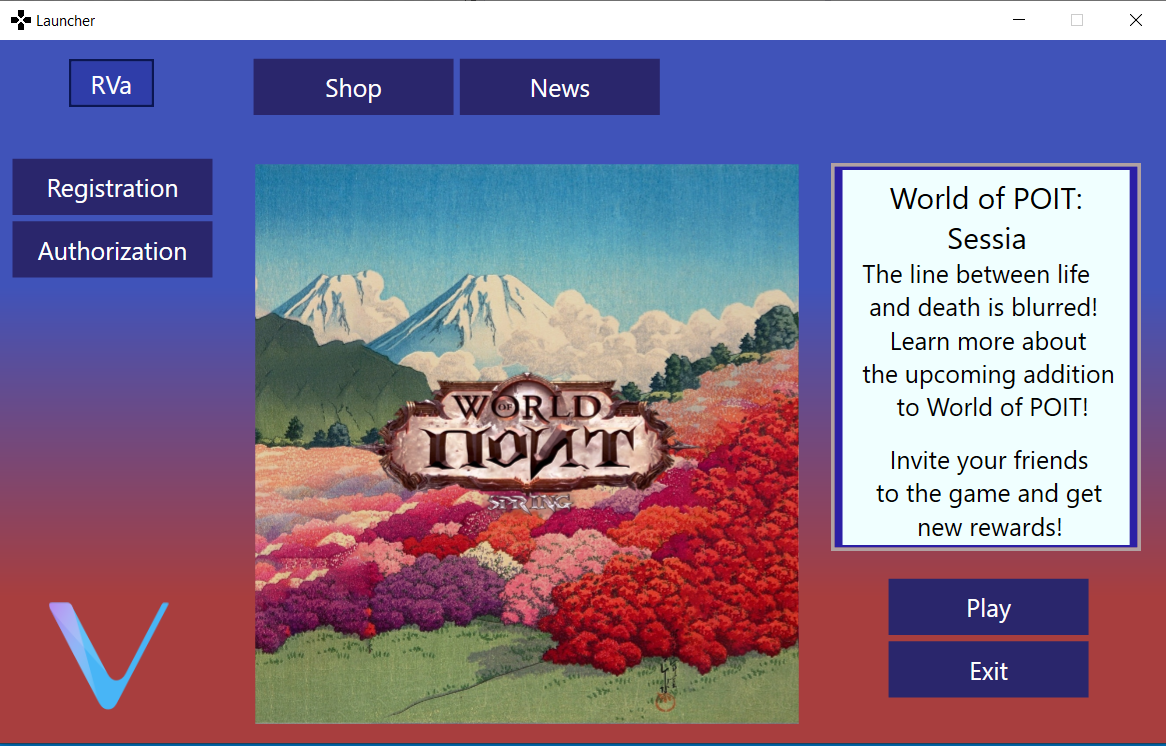


Рисунок 4.3 – Начальная страница

У пользователя доступны вкладки Registration, Authorization, Shop, News, Play и Exit. При переходе во вкладку Shop открывается форма с магазином (Рисунок 4.4).



Рисунок 4.4 – Окно магазина

Пользователю доступно окно новостей игрового лаунчера (Рисунок 4.5).

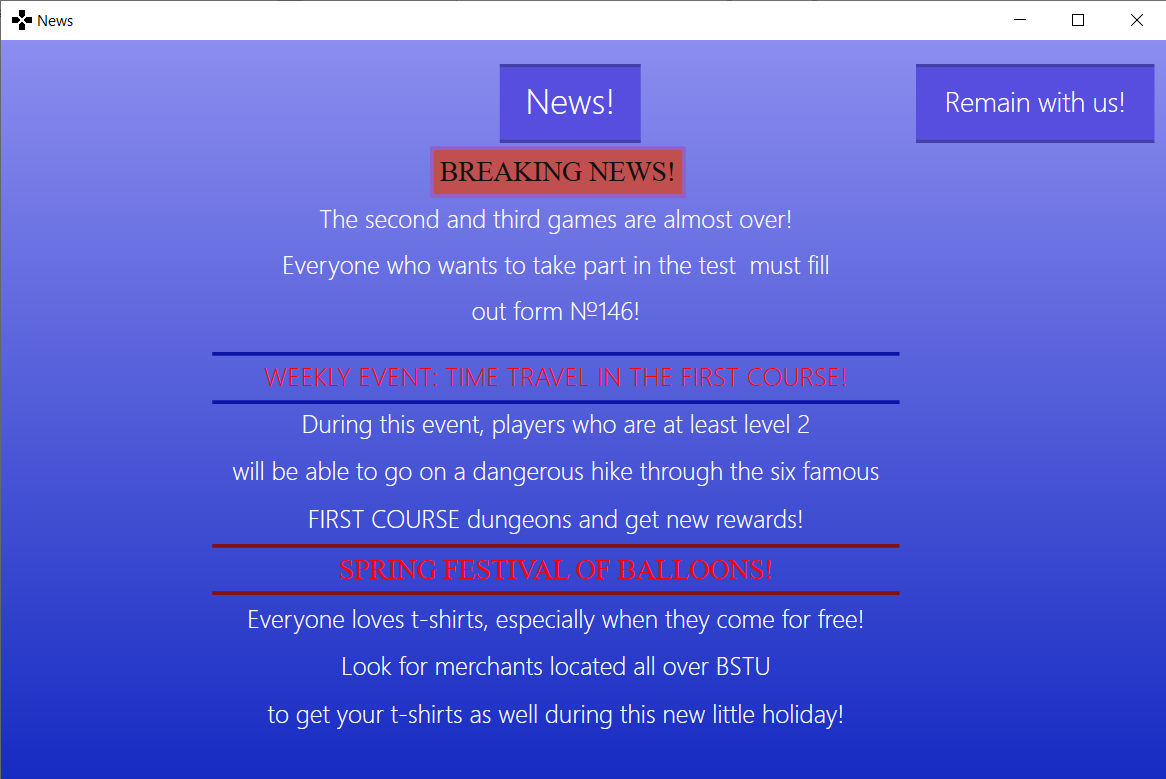


Рисунок 4.5 – Окно новостей

При переходе во вкладку Play открывается проект с игрой (Рисунок 4.6).

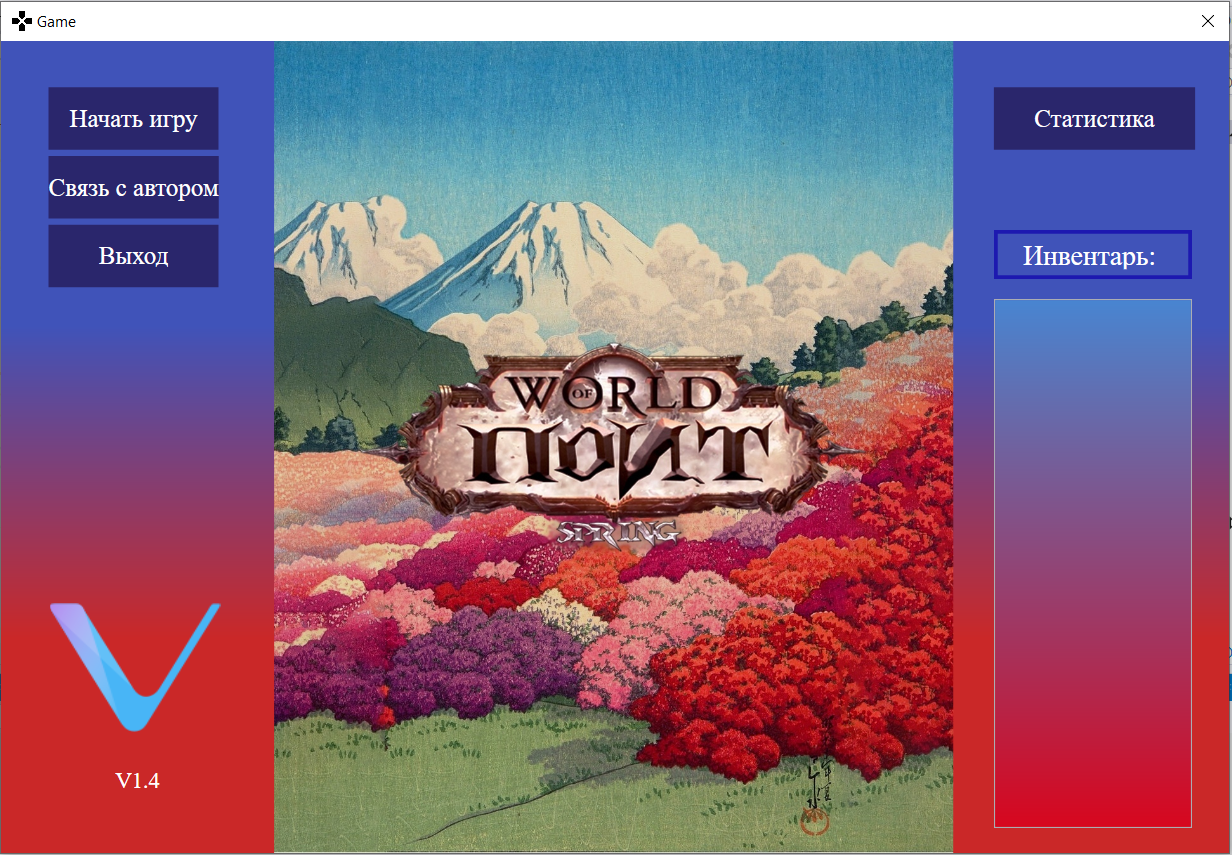


Рисунок 4.6 – Окно игры

Пользователь внутри игры может связаться с автором при нажатии на кнопку «Связь с автором». Открывается новое окно, в котором находятся контакты автора (Рисунок 4.7).



Рисунок 4.7 – Окно связи с автором

Так же пользователь может просматривать игровую статистику в процессе прохождения игры, для этого необходимо нажать на кнопку «Статистика» (Рисунок 4.8).

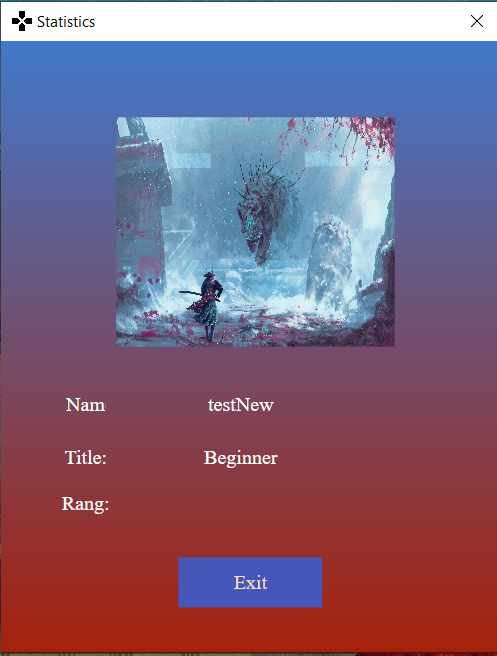


Рисунок 4.8 – Окно статистики

При входе в приложение под профилем администратора открывается

Соответствующее окно с таблицами для обработки поступивших запросов от пользователя (Рисунок 4.9).

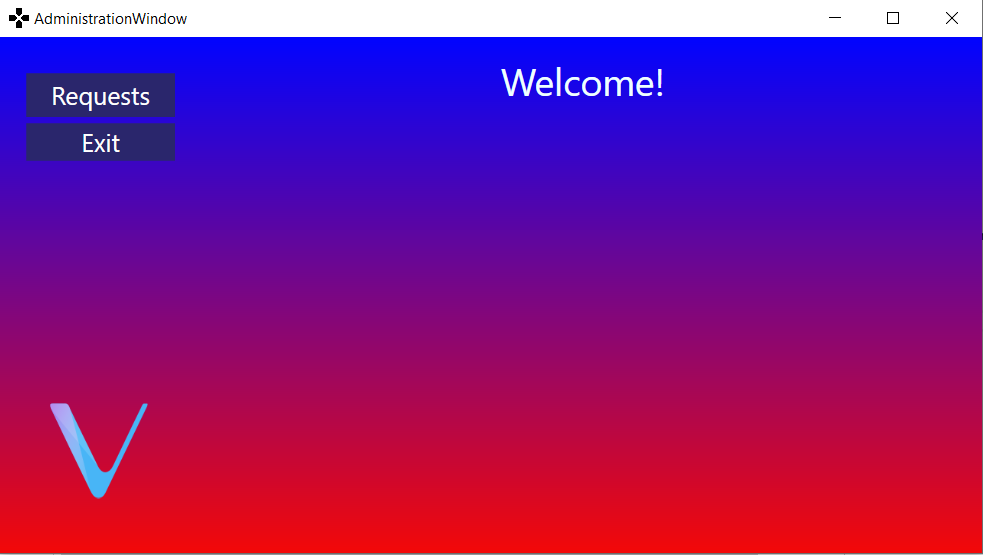


Рисунок 4.9 – Начальная страница для администратора

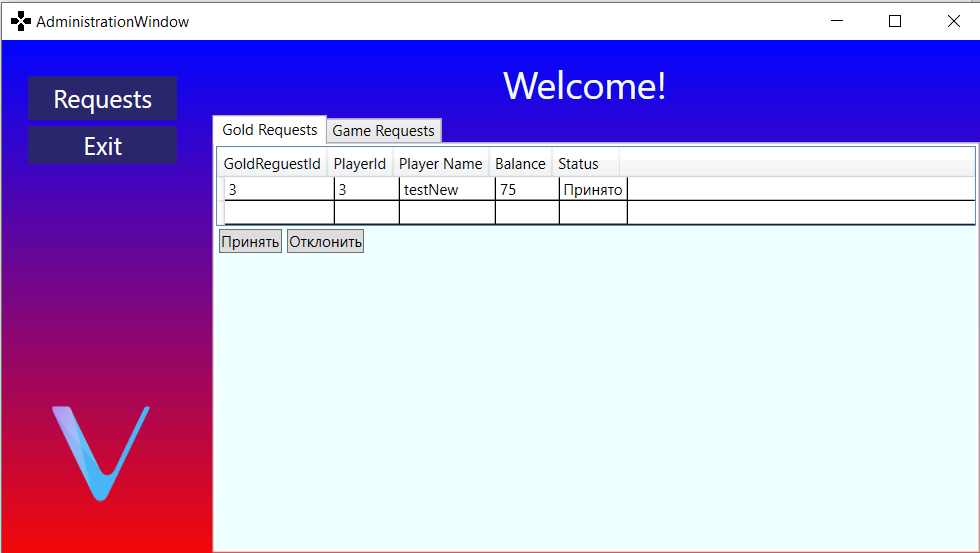


Рисунок 4.10 – Окно обработки запросов на покупку золота

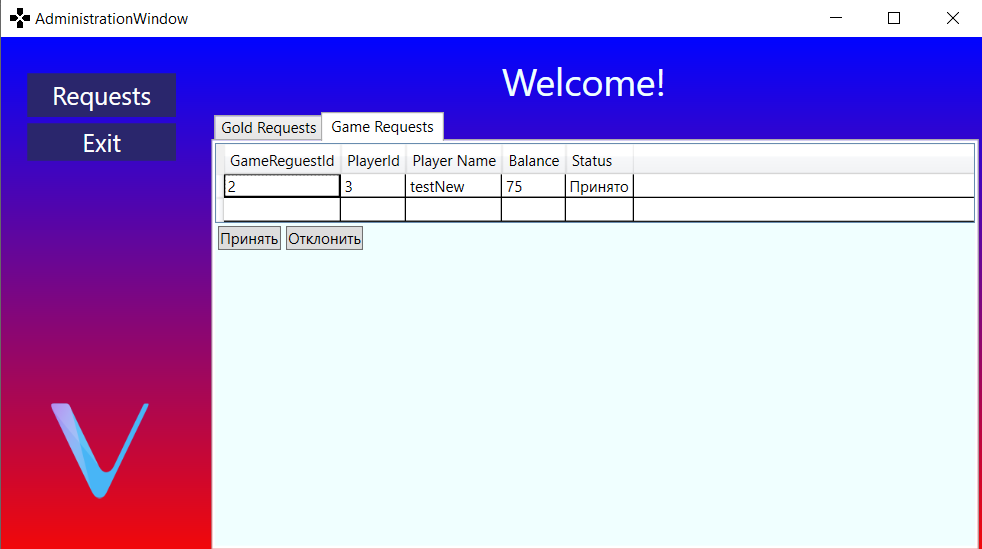


Рисунок 4.13 – Окно обработки запросов на покупку игры

**Заключение**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Романицкий В. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Заключение

Лит.

Листов

1

БГТУ

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных «Игровой Агрегатор», которая в купе с приложением формирует полноценное desktop-приложение. Основной целью курсового проекта стало проектирование базы данных для дальнейшей интеграции с приложением, которое помогло облегчить взаимодействие с базой данных посредством программного интерфейса. При разработке выполнены следующие пункты:

- авторизация и регистрация пользователей;

- разделение пользователя на админа и пользователя;

- покупка игры и игровой валюты;

- предзаказ игры;

- изменение игрового имени и пароля;

- хранение достижений пользователя;

- технология резервного копирования и восстановления, а также технология работы с мультимедийными типами данных;

Данный проект сложно сравнить с аналогами, каждый игровой агрегатор разрабатывается компаниями со своими целями и особенностями. Данный проект является показательным примером базы данных и приложения, позволяя охватить необходимый минимум требований для их использования.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

**Список литературы**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Список литературы

Лит.

Листов

1

БГТУ

1. Общие сведения о Microsoft Visual Studio [Электронный ресурс]. / Режим доступа <https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019>/

2. Интернет ресурс для проектирования и создания UML-диаграмм – https://app.diagrams.net

3. Microsoft Docs – https://docs.microsoft.com/en-us/sql/relational-databases/import-export/examples-of-bulk-import-and-export-of-xml-documents-sql-server?view=sql-server-ver15

4. Сайт professorweb.ru – https://professorweb.ru/my/sql-server/2012/level3/3\_5.php

5. Microsoft Docs – https://docs.microsoft.com/en-us/sql/t-sql/functions/openrowset-transact-sql?view=sql-server-ver15

6. Сайт info-comp.ru – https://info-comp.ru/sisadminst/486-full-text-search-ms-sql-server.html

7. METANIT – https://metanit.com/sharp/adonet/1.1.php

8. METANIT SQL – https://metanit.com/sql/.

9. METANIT C# – https://metanit.com/sharp/.

10. MSDN C# – <https://docs.microsoft.com/ru-ru/dotnet/csharp/>.

11. Microsoft Docs - https://docs.microsoft.com/ru-ru/sql/relational-databases/backup-restore/back-up-and-restore-of-sql-server-databases?view=sql-server-ver15

**Приложение А**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

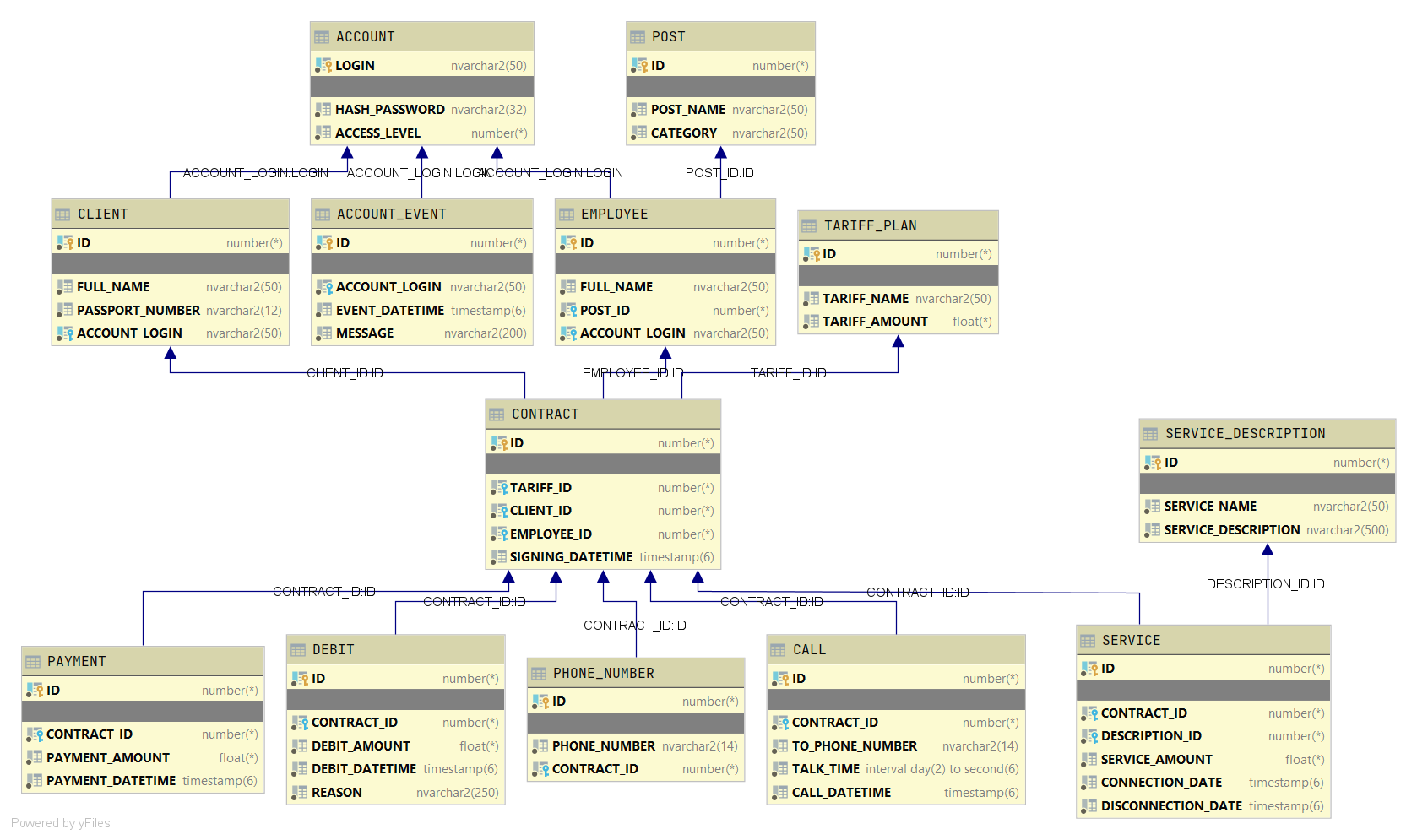
Приложение А

Лит.

Листов

1

БГТУ



**Приложение Б**

CREATE TABLE Account (  
login nvarchar2(50) PRIMARY KEY,  
hash\_password nvarchar2(32) NOT NULL,  
access\_level int NOT NULL  
);  
  
CREATE SEQUENCE account\_event\_seq INCREMENT BY 1;  
  
CREATE TABLE Account\_event (  
id int DEFAULT account\_event\_seq.*NEXTVAL* PRIMARY KEY,  
account\_login nvarchar2(50) NOT NULL,  
event\_datetime timestamp NOT NULL,  
message nvarchar2(200) NOT NULL,  
FOREIGN KEY (account\_login) REFERENCES Account(login)  
);  
  
CREATE SEQUENCE post\_seq INCREMENT BY 1;  
  
CREATE TABLE Post (  
id int DEFAULT post\_seq.*NEXTVAL* PRIMARY KEY,  
post\_name nvarchar2(50) NOT NULL,  
category nvarchar2(50) NOT NULL  
);  
  
CREATE SEQUENCE client\_seq INCREMENT BY 1;  
  
CREATE TABLE Client(  
id int DEFAULT client\_seq.*NEXTVAL* PRIMARY KEY,  
full\_name nvarchar2(50) NOT NULL,  
passport\_number nvarchar2(12) NOT NULL,  
account\_login nvarchar2(50) NOT NULL UNIQUE,  
FOREIGN KEY (account\_login) REFERENCES Account(login)  
);

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение Б

Лит.

Листов

1

БГТУ

CREATE SEQUENCE service\_description\_seq INCREMENT BY 1;  
  
CREATE TABLE Service\_description(  
id int DEFAULT service\_description\_seq.*NEXTVAL* PRIMARY KEY,  
service\_name nvarchar2(50) NOT NULL,  
service\_description nvarchar2(500) NOT NULL  
);  
  
CREATE SEQUENCE tariff\_plan\_seq INCREMENT BY 1;  
  
CREATE TABLE Tariff\_plan(  
id int DEFAULT tariff\_plan\_seq.*NEXTVAL* PRIMARY KEY,  
tariff\_name nvarchar2(50) NOT NULL,  
tariff\_amount float(126) NOT NULL  
);CREATE SEQUENCE employee\_seq INCREMENT BY 1;  
  
CREATE TABLE Employee(  
id int DEFAULT employee\_seq.*NEXTVAL* PRIMARY KEY,  
full\_name nvarchar2(50) NOT NULL,  
post\_id int NOT NULL,  
account\_login nvarchar2(50) NOT NULL UNIQUE,  
FOREIGN KEY (post\_id) REFERENCES Post(id),  
FOREIGN KEY (account\_login) REFERENCES Account(login)  
);CREATE SEQUENCE contract\_seq INCREMENT BY 1;  
  
CREATE TABLE Contract (  
id int DEFAULT contract\_seq.*NEXTVAL* PRIMARY KEY,  
tariff\_id int NOT NULL,  
client\_id int NOT NULL,  
employee\_id int NOT NULL,  
signing\_datetime timestamp NOT NULL,  
FOREIGN KEY (tariff\_id) REFERENCES Tariff\_plan(id),  
FOREIGN KEY (client\_id) REFERENCES Client(id),  
FOREIGN KEY (employee\_id) REFERENCES Employee(id)  
);  
  
CREATE SEQUENCE phone\_number\_seq INCREMENT BY 1;

CREATE TABLE Phone\_number (  
id int DEFAULT phone\_number\_seq.*NEXTVAL* PRIMARY KEY,  
phone\_number nvarchar2(14) NOT NULL,  
contract\_id int NOT NULL,  
CONSTRAINT phone\_regex\_pn CHECK (*REGEXP\_LIKE* (phone\_number, '^((\+\d{3})|(\d{2}))\d{2}\d{3}\d{4}$')),  
FOREIGN KEY (contract\_id) REFERENCES Contract(id) ON DELETE CASCADE  
);  
  
CREATE SEQUENCE call\_seq INCREMENT BY 1;  
  
CREATE TABLE Call (  
id int DEFAULT call\_seq.*NEXTVAL* PRIMARY KEY,  
contract\_id int NOT NULL,  
to\_phone\_number nvarchar2(14) NOT NULL,  
talk\_time interval day to second NOT NULL,  
call\_datetime timestamp NOT NULL,  
CONSTRAINT phone\_regex\_c CHECK (*REGEXP\_LIKE* (to\_phone\_number, '^((\+\d{3})|(\d{2}))\d{2}\d{3}\d{4}$')),  
FOREIGN KEY (contract\_id) REFERENCES Contract(id) ON DELETE CASCADE  
);  
  
CREATE SEQUENCE payment\_seq INCREMENT BY 1;  
  
CREATE TABLE Payment(  
id int DEFAULT payment\_seq.*NEXTVAL* PRIMARY KEY,  
contract\_id int NOT NULL,  
payment\_amount float(126) NOT NULL,  
payment\_datetime timestamp NOT NULL,  
FOREIGN KEY (contract\_id) REFERENCES Contract(id) ON DELETE CASCADE  
);  
  
CREATE SEQUENCE debit\_seq INCREMENT BY 1;  
  
CREATE TABLE Debit(  
id int DEFAULT debit\_seq.*NEXTVAL* PRIMARY KEY,  
contract\_id int NOT NULL,  
debit\_amount float(126) NOT NULL,  
debit\_datetime timestamp NOT NULL,  
reason nvarchar2(250) NOT NULL,  
FOREIGN KEY (contract\_id) REFERENCES Contract(id) ON DELETE CASCADE  
);CREATE SEQUENCE service\_seq INCREMENT BY 1;  
  
CREATE TABLE Service (  
id int DEFAULT service\_seq.*NEXTVAL* PRIMARY KEY,  
contract\_id int NOT NULL,  
description\_id int NOT NULL,  
service\_amount float(126) NOT NULL,  
connection\_date timestamp NOT NULL,  
disconnection\_date timestamp NOT NULL,  
FOREIGN KEY (contract\_id) REFERENCES Contract(id) ON DELETE CASCADE,  
FOREIGN KEY (description\_id) REFERENCES Service\_description(id)  
);

**Приложение В**

ФИО

Подпись

Дата

Лист

1

КП

Разраб.

Бельский А. А.

Провер.

Блинова Е. А.

Н. контр.

Утверд.

Приложение В

Лит.

Листов

1

БГТУ

|  |  |  |
| --- | --- | --- |
| Название процедуры | Описание | Принимаемые параметры |
| Registration | Процедура регистрации пользователя | @UserName, @Password |
| Add\_Gold | Процедура добавления золота пользователю | @playerId, @Input\_gold |
| Accept\_gold | Процедура добавления золота пользователю | @RequestId, @PlayerId, @Gold |
| Accept\_game | Процедура добавления игры пользователю | @RequestId, @PlayerId |
| ChangeInformation | Процедура редактирования данных пользователя | @UserId, @NewUserName, @NewUserPassword |
| PreorderProcedure | Процедура предзаказа игры | @PlayerId |
| Save\_Statistics | Процедура для сохранения статистики | @PlayerId, @title, @rang |
| Add100kToUser | Процедура добавления 100000 строк в таблицу Users | @UserId, @UserPassword, @IsAdmin, @UserName |