Министерство образования Республики Беларусь

### Учреждение образования «[Институт информационных технологий](https://www.google.by/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&sqi=2&ved=0CDcQFjAC&url=http%3A%2F%2Fwww.iit-bsuir.by%2F&ei=unOvUpDmGemB4gSZ3IGQDQ&usg=AFQjCNHEr4the3QhkSSjmxbzcNJBZi5-Tg&sig2=GxV3hq7_8t34Csduk0HElg&bvm=bv.57967247,d.bGE&cad=rja) Белорусского государственного университета информатики и радиоэлектроники»

Факультет компьютерных технологий

Кафедра информационных систем и технологий

Дисциплина: Базы данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

«Программное средство документооборота Интернет-провайдера»

БГУИР КП 1-40 01 01 5810027 ПЗ

Студент: гр.581061 Фут Д.С.

Руководитель: Бакунов А.М.

Минск 2017

**СОДЕРЖАНИЕ**

Введение 4

1 Анализ предметной области и постановка задачи 5

1.1 Анализ существующих аналогов 5

1.2 Постановка задачи 5

2 Входные и выходные данные 6

2.1 Входные данные 6

2.2 Выходные данные 6

3 Физическая и логическая модель данных. Схема данных. 7

4 Выбор и обоснование средств разработки 9

4.1 Разработка спецификации требований 9

4.2 Разработка архитектуры программного средства 9

4.3 Обоснование выбора языка программирования 10

4.4 Обоснование выбора среды разработки 10

5 Тестирование и отладка ПС 12

6 Руководство пользователя 15

Заключение 24

Список использованных источников 25

Приложение А. Листинг программы 26

Приложение B. Диаграмма классов 39

Приложение С. Диаграмма вариантов использования 40

**ВВЕДЕНИЕ**

Возрастающие темпы информатизации общества повышают значение вычислительной техники в управленческих процессах. Использование возможностей современной вычислительной техники для автоматизации процесса обработки информации позволяет увеличить производительность труда, повысить эффективность работы с документами и ускорить обмен управленческой информацией. Предприятия активно используют вычислительную технику для ведения бухгалтерского учета, контроля за выполнением заказов и договоров, подготовки деловых документов.

Создание автоматизированных рабочих мест позволяет эффективно обрабатывать большие потоки информации, которые имеют определенную структуру, зависящую от особенностей места применения. Это позволяет осуществлять индивидуальный подход к автоматизации именно тех функций, которые выполняются данным подразделением. Введение на предприятии автоматизированных рабочих мест позволяет значительно сократить время выполнения работ и повысить их точность, облегчить труд специалистов.

Основной задачей этого курсового проекта является разработка программного средства для документооборота Интернет-провайдера. Данное программное средство должно помогать администраторам и менеджерам документооборота контролировать процесс работы с документами и клиентами.

**1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ**

**1.1 Анализ существующих аналогов**

На данный момент времени существует множество аналогов программных средств по автоматизации рабочих мест сотрудников.

К основным недостаткам большинства таких программ можно отнести то, что для каждого рабочего места требуются свои функции, в зависимости от потребностей заказчика, соответственно программа, подходящая одному, не всегда может подойти другому, исходя из вышеизложенного причина создания программного средства обусловлена индивидуальными требованиями заказчика и невозможностью найти аналог под такие требования.

**1.2 Постановка задачи**

Анализ литературы и предметной области показали, что реализуемое программное средство имеет большое количество аналогов, но каждый из которых имеет свои недостатки.

Основной целью является – разработка программного средства для управления документооборотом Интернет-провайдера.

В данном программном средстве необходимо реализовать:

* Учет клиентов провайдера. Для каждого клиента необходимо хранить: имя, фамилию, отчество, телефон, почтовый адрес, адрес проживания, номер контракта;
* Учет документов. Для каждого документа необходимо хранить: ссылку на документ, дату создания в системе, принадлежность клиенту, принадлежность менеджеру, статус документа, тип документа;
* Учёт пользователей системы. Для каждого пользователя необходимо хранить: логин, пароль, фамилию, имя, отчество, телефон, почтовый адрес и информацию о том, является ли пользователь администратором;
* Прямой доступ к взаимодействию с базой данных;
* Возможность ручного создания бэкапа базы данных и возможность восстановить базу данных с помощью бэкапа.

**2 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ**

**2.1 Входные данные**

Входной информацией является:

* информация о клиенте;
* информация о документе;
* информация о пользователе;
* информация о статусе документа;
* информация о типе документа.

**2.2 Выходные данные**

К выходной информации относится:

* список всех клиентов;
* список документов;
* список пользователей;
* список статусов документа;
* список типов документа.

**3 ФИЗИЧЕСКАЯ И ЛОГИЧЕСКАЯ МОДЕЛЬ ДАННЫХ. СХЕМА ДАННЫХ.**

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

База данных соответствует реляционной модели данных, где каждый выделенный в ходе проектировании сущности соответствует таблица.

Логическая модель хранилища данных представляет собой описание сущностей, атрибутов и отношений между ними. На рисунке 3.1 представлена логическая модель данных.

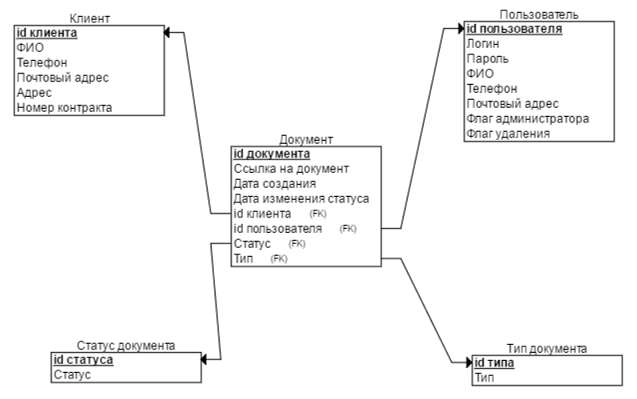


Рисунок 3.1 – Логическая модель данных

Физическая модель данных описывает реализацию объектов логической модели на уровне объектов конкретной базы данных, уточняя конкретные названия полей и их типы данных. На рисунке 3.2 представлена физическая модель данных.

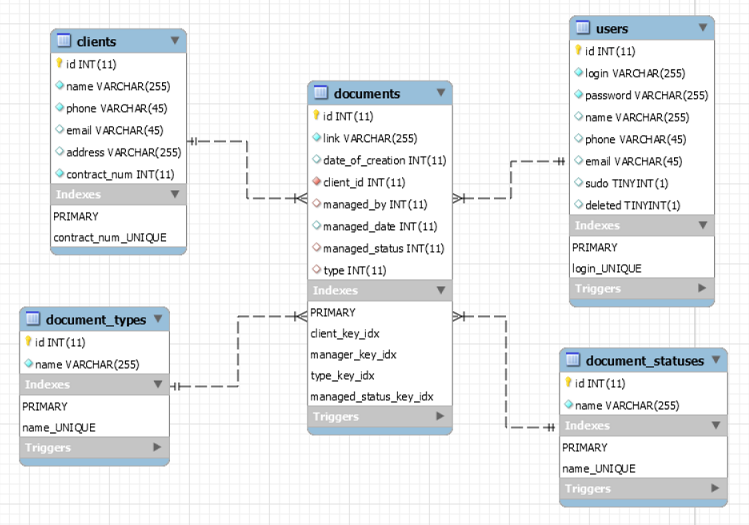


Рисунок 3.2 – Физическая модель данных

**4 ВЫБОР И ОБОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ**

**4.1 Разработка спецификаций требований**

Программа будет разрабатываться, и функционировать в операционной системе Windows. В версиях от Windows xp до Windows 10.

Windows — семейство проприетарных операционных систем корпорации Microsoft, ориентированных на применение графического интерфейса при управлении. Изначально Windows была всего лишь графической надстройкой для MS-DOS. По состоянию на август 2014 года под управлением операционных систем семейства Windows по данным ресурса Net Applications работает около 89% персональных компьютеров. Это и было основание выбора данной операционной системы как основной рабочей системы проекта.

Минимальные требования:

* + - процессор: Intel Pentium II 400 MHz или аналогичный;
    - оперативная память: 512 Mб;
    - видеокарта: объем 256 Mб;
    - операционная система: Windows XP;
    - устройства ввода: клавиатура, мышь;
    - устройство вывода: монитор.

**4.2 Разработка архитектуры программного средства**

Программное средство должно будет хранить все имеющуюся информацию в базе данных. Для хранения информации будет использоваться база данных использующая реляционную модель.

В реляционной СУБД все обрабатываемые данные представляются в виде плоских таблиц. Информация об объектах определенного вида представляется в табличном виде: в столбцах таблицы сосредоточены различные атрибуты объектов, а строки предназначены для сведения описаний всех атрибутов к отдельным экземплярам объектов.

В качестве самой СУБД будет использоваться MySQL.

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

**4.3 Обоснование выбора языка программирования**

Java представляет собой язык программирования и платформу вычислений, которая была впервые выпущена Sun Microsystems в 1995 г.

Существует множество приложений и веб-сайтов, которые не работают при отсутствии установленной Java, и с каждым днем число таких веб-сайтов и приложений увеличивается. Java отличается быстротой, высоким уровнем защиты и надежностью. От портативных компьютеров до центров данных, от игровых консолей до суперкомпьютеров, используемых для научных разработок, от сотовых телефонов до сети Интернет.

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее.

В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода. В Java добавлены все хорошие черты C++.

Три ключевых элемента объединились в технологии языка Java

Java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML

Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты

Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов

**4.4 Обоснование выбора среды разработки**

IntelliJ IDEA — интегрированная среда разработки программного обеспечения на многих языках программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Первая версия появилась в январе 2001 года и быстро приобрела популярность, как первая среда для Java с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди прочих возможностей, среда хорошо совместима со многими популярными свободными инструментами разработчиков, такими как CVS, Subversion, Apache Ant, Maven и JUnit. В феврале 2007 года разработчики IntelliJ анонсировали раннюю версию плагина для поддержки программирования на языке Ruby.

Начиная с версии 9.0, среда доступна в двух редакциях: Community Edition и Ultimate Edition. Community Edition является полностью свободной версией, доступной под лицензией Apache 2.0, в ней реализована полная поддержка Java SE, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. В редакции Ultimate Edition реализована поддержка Java EE, UML-диаграмм, подсчёт покрытия кода, а также поддержка других систем управления версиями, языков и фреймворков.

**5 ТЕСТИРОВАНИЕ И ОТЛАДКА ПС**

Тесты существенно различаются по задачам, которые с их помощью решаются, и по используемой технике. Различие задач тестирования приводит, естественным образом, к необходимости использовать весьма разнообразные типы (виды) тестирования. Принято подразделять тестирование на виды по следующим категориям:

* по объектам (элементам) тестирования, часто разделение на виды тестов по данному критерию называют разделением тестирования на уровни;
* по глубине тестирования, то есть разделение тестовых испытаний на типы проводится в зависимости от количества времени и объема тестируемых компонент программного продукта.

1. Модульное тестирование (Автономное или Unit-тестирование).

На данном уровне тестируются по отдельности небольшие элементы системы, максимально отделенные от других элементов и, в то же время, пригодные для тестирования.

2. Комплексное тестирование (Сборочное тестирование, integration testing или interface testing).

На данном уровне тестируются объединенные элементы (компоненты или подсистемы) общей системы, чаще всего некоторая взаимодействующая между собой группа элементов

3. Системное тестирование (system testing).

После того, как система собрана из составляющих компонентов, она должна быть протестирована на соответствие “Системным спецификациям” – реализованы ли все функциональные и нефункциональные требования к разрабатываемой системе.

4. Приемочное тестирование (Приемо-сдаточное тестирование или acceptance testing).

На данном уровне завершенное приложение (система) тестируется Заказчиком, конечными пользователями или соответствующими уполномоченными с целью определения соответствия системы “Требованиям Заказчика” и готовности системы к внедрению

5. Операционное тестирование (Release Testing)

Тестирование в среде эксплуатации позволяет выявить и нефункциональные проблемы, такие как: конфликт с другими системами, смежными в области бизнеса или в программных и электронных окружениях; недостаточная производительность системы в среде эксплуатации и др.

Виды тестирования:

Инсталляционное тестирование (Installation testing). Основная цель состоит в том, чтобы убедиться, что продукт может быть установлен/деинсталлирован при различных условиях – таких как: новая инсталляция, усовершенствование системы (upgrade), установка по умолчанию, полная установка, установка по выбору.

Дымное тестирование (проверка на дым, Smoke testing). Как правило, используется для определения, готова ли программа для проведения более обширного тестирования.

Функциональное тестирование (Functional testing). Проверка соответствия продукта функциональным требованиям и спецификациям.

Функциональные требования включают в себя: Функциональную пригодность, точность, способность к взаимодействию, соответствие стандартам и правилам, защищённость.

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. Тестирование нефункциональных требований, то есть: тестирование производительности, защищенности, использования и восстановления.

Тестирование графического интерфейса пользователя (User Interface testing). Тестирование интерфейса – экранов, кнопок и т.д. Большая часть функциональности ПО реализуется, как правило, через пользовательский интерфейс.

Для начала проверим приложение на безопасность. Попробуем выполнить вход в систему с ошибочными данными.

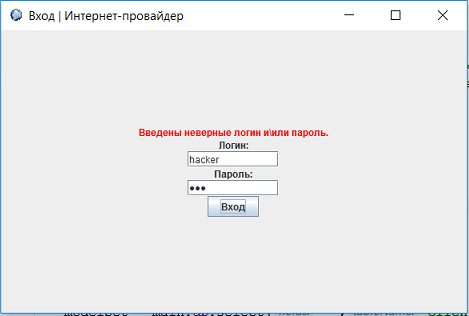


Рисунок 5.1 – Реакция приложения на ввод некорректных данных

Попробуем удалить статус, который в данный момент используется в системе. В случае если это удастся, то появятся некорректная информация в системе.

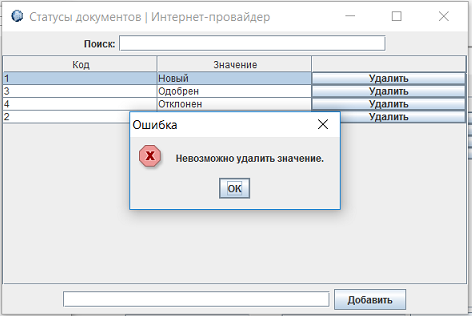


Рисунок 5.2 – Реакция приложения на попытку удаления используемого статуса

Далее попробуем добавить статус, который уже есть в системе. Если это удастся, то в приложении будет иметь место дублирование уникальной информации, что будет явной ошибкой в системе.

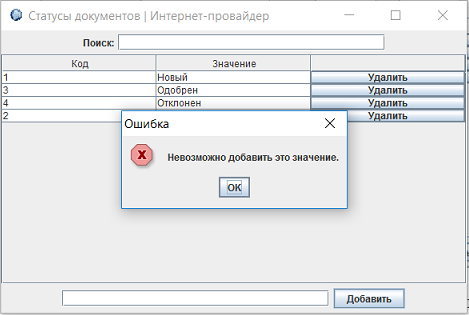


Рисунок 5.3 – Реакция приложения на попытку дублировать существующее значение

Приложение удачно прошло этап тестирования, имеет хороший уровень безопасности и не допускает пользователю внести некорректную информацию в базу.

**6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

После запуска приложения можно увидеть главную форму входа. На ней размещены элементы для ввода логина, пароля и кнопка «Вход».

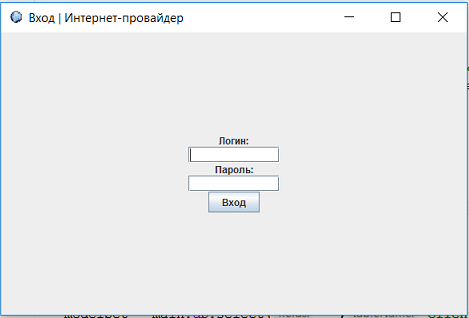


Рисунок 6.1 – Форма входа в приложение

После нажатия на кнопку «Вход» перед пользователем появиться форма, на которой расположены кнопки, позволяющие ему начать работу с нужным разделом системы. Для всех пользователей это будут кнопки: «Клиенты», «Мои документы». Для администраторов также появятся кнопки: «Менеджеры», «Статусы документов», «Типы документов», «Все документы», «Ручные запросы», «Работа с бэкапами».

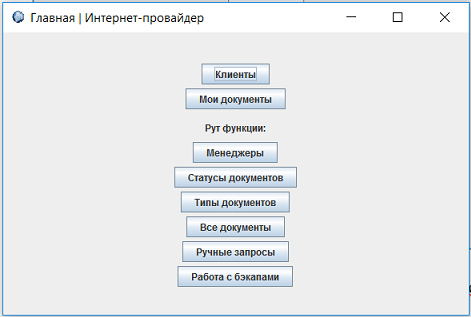


Рисунок 6.2 – Главная форма приложения для администратора

При нажатии одну из кнопок «Статусы документов», «Типы документов» откроется стандартная форма редактирования справочной сущности. Все они имеют одинаковый вид и функционал. Продемонстрирую функционал на форме «Типы документов».

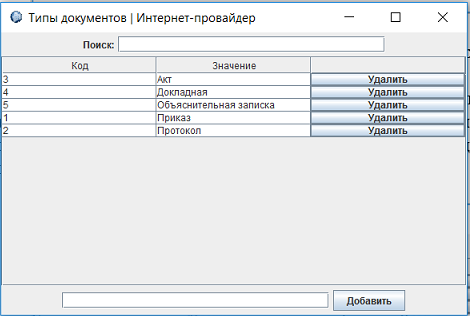


Рисунок 6.3 – Форма для управления типами документов

На этой форме можно осуществить изменение значения путём редактирования его прямо в таблице. Также возможно удалить и добавить новое значение. В верхней части формы доступно поле для фильтрации списка по полю название.

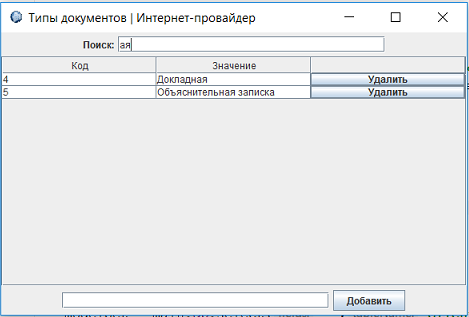


Рисунок 6.4 – Список типов при активной фильтрации

Для управления менеджерами, пользователь может нажать на кнопку «Менеджеры» и приложение выдаст ему форму для управления.

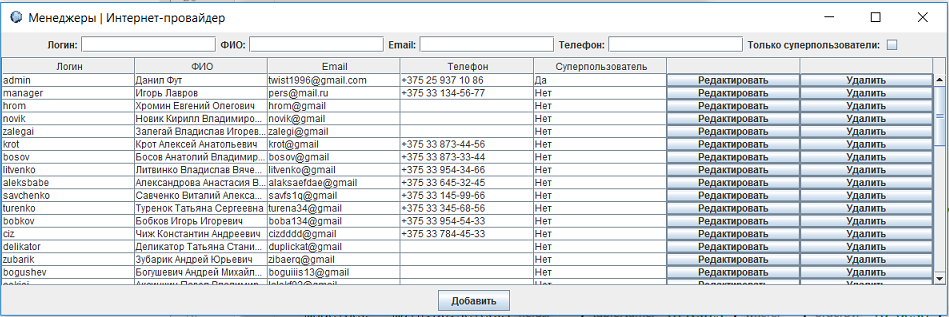


Рисунок 6.5 – Форма для управления менеджерами

На данной форме можно получить доступ к форме редактирования, добавления пользователя, удалить пользователя или использовать фильтрацию. При нажатии на кнопку «Добавить» откроется форма добавления нового пользователя.

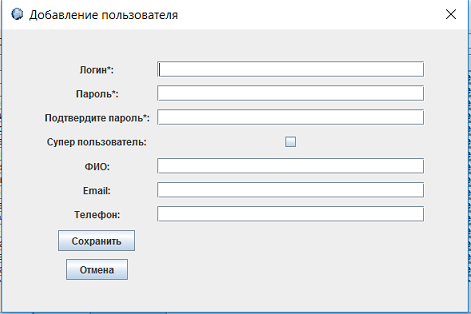


Рисунок 6.6 – Форма добавления пользователя

Для управления клиентами, пользователь может нажать на кнопку «Клиенты» и приложение выдаст ему форму для управления.

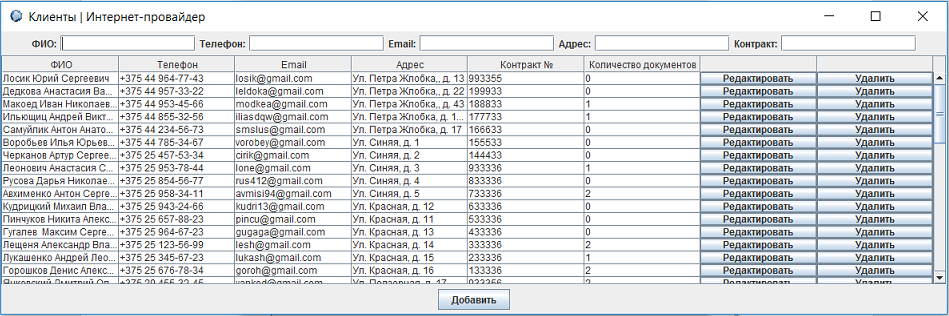


Рисунок 6.7 – Форма для управления клиентами

На данной форме можно получить доступ к форме редактирования, добавления клиента, удалить клиента или использовать фильтрацию. При нажатии на кнопку «Редактировать» откроется форма редактирование информации о клиенте.

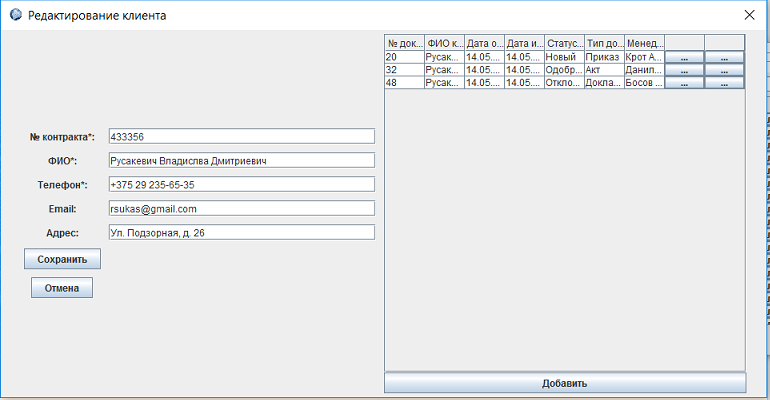


Рисунок 6.8 – Форма редактирования клиента

Для управления документами, пользователь может нажать на кнопку «Мои документы» и приложение выдаст ему форму для управления.

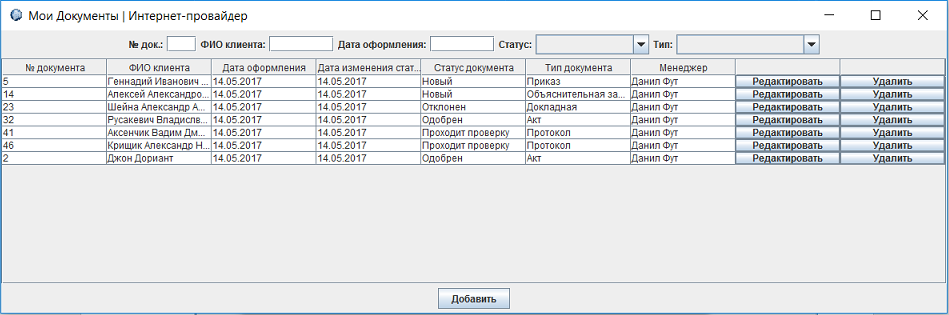


Рисунок 6.9 – Форма для управления документами

На данной форме можно получить доступ к форме редактирования, добавления документа, удалить документ или использовать фильтрацию. При нажатии на кнопку «Редактировать» откроется форма редактирования информации о документе.

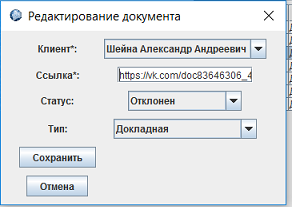


Рисунок 6.10 – Форма редактирования информации о документе

Для прямого доступа к работе с базой, пользователь может нажать на кнопку «Ручные запрос» и приложение выдаст ему форму для работы с запросами.

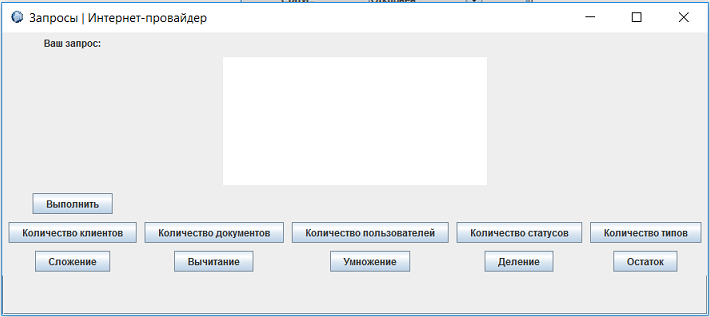


Рисунок 6.11 – Форма для составления запросов к базе данных

На данной форме есть кнопки для заготовленных запросов, которые автоматически заполнять поле «Ваш запрос» и позволять пользователю получить результат.

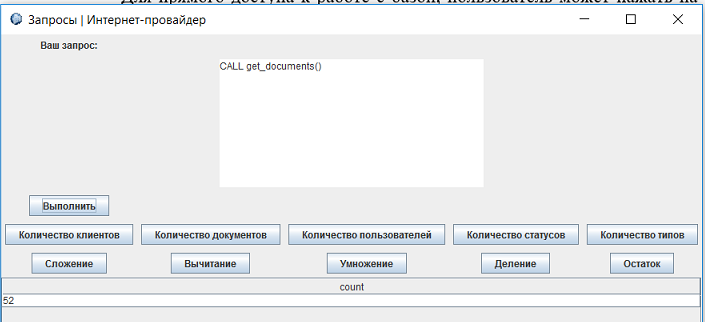


Рисунок 6.12 – Результат нажатия на кнопку «Количество документов»

Пользователь может ввести любой запрос в поле для запроса и получить результат. Если запрос направлен на выборку каких-либо данных, то программа выдаст ему таблицу с этими данными. Если запрос направлен на добавление, изменение, удаление каких-либо данных, то программа уведомит пользователя о результат запроса.

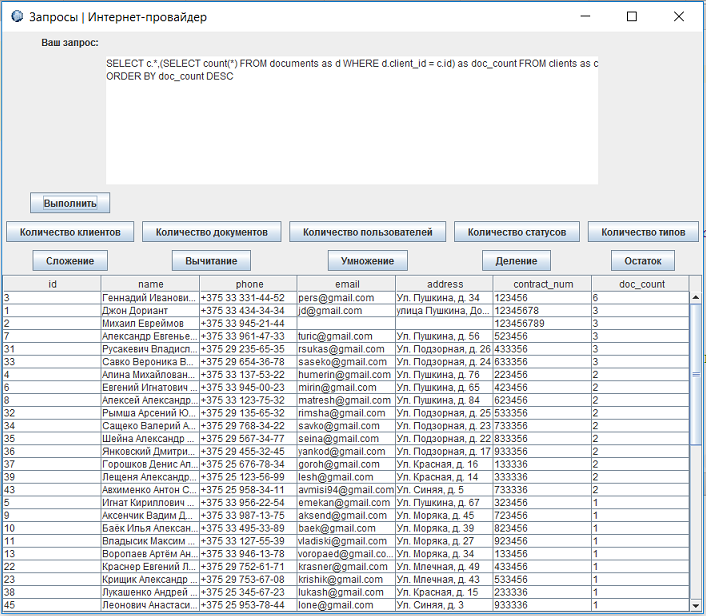


Рисунок 6.13 – Результат выполнения запроса на выборку

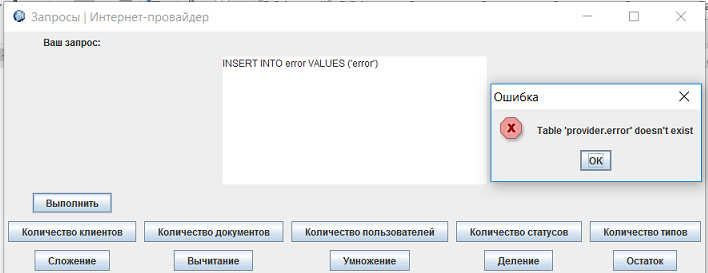


Рисунок 6.14 – Результат выполнения запроса с ошибкой

Для работы с бэкапами, пользователь может нажать на кнопку «Работа с бэкапами» и приложение выдаст ему форму для работы с бэкапами.

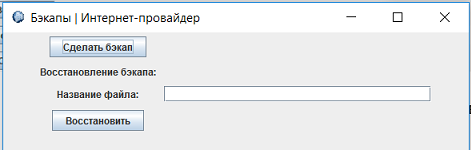


Рисунок 6.15 – Форма для работы с бэкапами

После нажатия на кнопку «Сделать бэкап» система оповестит пользователя об результате процесса.

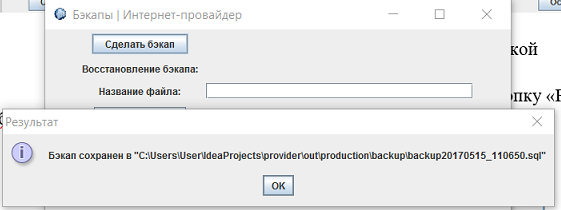


Рисунок 6.16 – Оповещение о результате создания бэкапа

Для восстановления базы с помощью существующего бэкапа, пользователь должен ввести название существующего бэкапа и нажать на кнопку «Восстановить».

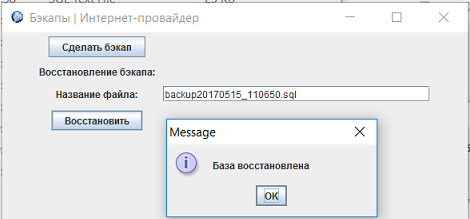


Рисунок 6.17 – Оповещение пользователя о результате восстановления

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте разработано программное средство для документооборота Интернет-провайдера.

В данном программном средстве реализовано:

* управление информацией о клиентах;
* управление информацией о документах;
* управление информацией о пользователях;
* доступ к прямой работе с базой данных;
* управление системой создания резервных копий.

Курсовая работа была реализована на языке программирования Java с использованием среды разработки JetBrains IDEA и библиотек графического интерфейса Swing и AWT.

Все задачи, которые ставились в начале проектирования были реализованы и в некоторых местах дополнены.

В дальнейшем программное средство может быть модифицировано различными элементами управления и настройками, которые позволят сделать данное программное средство более универсальным, а также позволят решать другие более сложные задачи чем простое управление СУБД.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Свободная энциклопедия [Электронный ресурс] / центр информ. Режим доступа: http://ru.wikipedia.org/wiki/Windows\_10.
2. Программирование на языке Java: учеб. пособие / В.В. Фаронов; под ред. В.С. Кустов. – Питер, 2007. – 240с.
3. Язык Java: самоучитель / А.В. Фролов, Г.В. Фролов; под ред. О.А Фаронов. – Москва, 2003. – 560с.
4. Дьюсон Робин. для начинающих разработчиков. СПб.: БХВ-Питер, 2009. – 704 с.
5. Сайт разработчика [Электронный ресурс] центр информ. Режим доступа: http://www.protesting.ru/testing/. Дата доступа: 04.04.2017.
6. Энциклопедия [Электронный ресурс] центр информ. Режим доступа: http://habrahabr.ru/post/169381/ /ARM. Дата доступа: 06.04.2017.
7. Канер С., Фолк Дж., Енг Кек Нгуен. Тестирование программного обеспечения, 2004. – 73 с.

**ПРИЛОЖЕНИЕ А.**

Листинг программы

**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.security.MessageDigest;  
**import** java.security.NoSuchAlgorithmException;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
  
*/\*\*  
 \* Created by User on 16.04.2017.  
 \*/***public class** AuthForm {  
 **static void** showForm() {  
  
 JFrame startFrame = **new** JFrame(**"Вход | Интернет-провайдер"**);  
 startFrame.setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
 **try** {  
 startFrame.setIconImage(ImageIO.*read*(**new** File(**"out/production/provider/images/diamond\_ico.jpg"**)));  
 } **catch** (IOException ie) {  
 ie.printStackTrace();  
 }  
  
 startFrame.setSize(600,400);  
 startFrame.setLocationRelativeTo(**null**);  
  
 JPanel loginPanel = **new** JPanel(**new** GridBagLayout());  
 GridBagConstraints c = **new** GridBagConstraints();  
 c.**gridx** = 1;  
  
 JLabel errorFieldLabel = **new** JLabel();  
 errorFieldLabel.setForeground(Color.***RED***);  
 JLabel loginFieldLabel = **new** JLabel(**"Логин:"**);  
 JTextField loginField = **new** JTextField(**""**,10);  
 JLabel passwordFieldLabel = **new** JLabel(**"Пароль:"**);  
 JPasswordField passwordField = **new** JPasswordField(**""**,10);  
 JButton submitButton = **new** JButton(**"Вход"**);  
  
 submitButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 ResultSet loggedSet = **null**;  
  
 errorFieldLabel.setText(**""**);  
 String login = loginField.getText();  
 String password = String.*valueOf*(passwordField.getPassword());  
  
 **if** ((login.length() > 0) && (password.length() > 0)) {  
 **try** {  
 loggedSet = main.*db*.select(**"\*"**,**"users"**,**"login = ? AND password = md5(?) AND deleted != 1"**,**new** String[]{login,password},**""**,**"1"**);  
  
  
 **while** (loggedSet.next()) {  
 main.*activeManager* = **new** Manager(loggedSet);  
  
 startFrame.setVisible(**false**);  
 startFrame.dispose();  
  
 main.*Backupdbtosql*();  
 GeneralForm.*showForm*();  
 **break**;  
 }  
 } **catch** (SQLException sqlEx) {  
 sqlEx.printStackTrace();  
 } **finally** {  
 main.*db*.closeStatementSet();  
 }  
  
 }  
  
 errorFieldLabel.setText(**"Введены неверные логин и\\или пароль."**);  
 }  
 });  
  
  
 loginPanel.add(errorFieldLabel,c);  
 loginPanel.add(loginFieldLabel,c);  
 loginPanel.add(loginField,c);  
 loginPanel.add(passwordFieldLabel,c);  
 loginPanel.add(passwordField,c);  
 loginPanel.add(submitButton,c);  
  
  
 startFrame.add(loginPanel,BorderLayout.***CENTER***);  
 startFrame.setVisible(**true**);  
  
 }  
}

**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.io.File;  
**import** java.io.IOException;  
  
*/\*\*  
 \* Created by User on 16.04.2017.  
 \*/***public class** GeneralForm {  
 **static void** showForm() {  
  
 JFrame startFrame = **new** JFrame(**"Главная | Интернет-провайдер"**);  
 startFrame.setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
  
 startFrame.setSize(600,400);  
 startFrame.setLocationRelativeTo(**null**);  
  
 **try** {  
 startFrame.setIconImage(ImageIO.*read*(**new** File(**"out/production/provider/images/diamond\_ico.jpg"**)));  
 } **catch** (IOException ie) {  
 ie.printStackTrace();  
 }  
  
 JPanel mainPanel = **new** JPanel(**new** GridBagLayout());  
 GridBagConstraints c = **new** GridBagConstraints();  
 c.**gridx** = 1;  
 Insets standardInset = **new** Insets(5,0,0,0);  
 c.**insets** = standardInset;  
  
 JButton clientsButton = **new** JButton(**"Клиенты"**);  
 clientsButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** ClientsForm();  
 }  
 });  
  
 JButton myOrdersButton = **new** JButton(**"Мои документы"**);  
 myOrdersButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** OrdersForm(main.*activeManager*.getId());  
 }  
 });  
  
 mainPanel.add(clientsButton,c);  
 mainPanel.add(myOrdersButton,c);  
  
 **if** (main.*activeManager*.getSudo()) {  
 JLabel sudoLabel = **new** JLabel(**"Рут функции:"**);  
  
 JButton managersButton = **new** JButton(**"Менеджеры"**);  
 managersButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** ManagersForm();  
 }  
 });  
  
 JButton ordersStatusesButton = **new** JButton(**"Статусы документов"**);  
 ordersStatusesButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** ListAddForm(**"Статусы документов"**,**"document\_statuses"**);  
 }  
 });  
  
 JButton typesButton = **new** JButton(**"Типы документов"**);  
 typesButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** ListAddForm(**"Типы документов"**,**"document\_types"**);  
 }  
 });  
  
 JButton ordersButton = **new** JButton(**"Все документы"**);  
 ordersButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** OrdersForm(0);  
 }  
 });  
  
 JButton queryButton = **new** JButton(**"Ручные запросы"**);  
 queryButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** QueryForm();  
 }  
 });  
  
 JButton backupButton = **new** JButton(**"Работа с бэкапами"**);  
 backupButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** BackupForm();  
 }  
 });  
  
 c.**insets** = **new** Insets(15,0,5,0);  
 mainPanel.add(sudoLabel,c);  
  
 c.**insets** = standardInset;  
 mainPanel.add(managersButton,c);  
 mainPanel.add(ordersStatusesButton,c);  
 mainPanel.add(typesButton,c);  
 mainPanel.add(ordersButton,c);  
 mainPanel.add(queryButton,c);  
 mainPanel.add(backupButton,c);  
 }  
  
 startFrame.add(mainPanel,BorderLayout.***CENTER***);  
 startFrame.setVisible(**true**);  
 }  
}

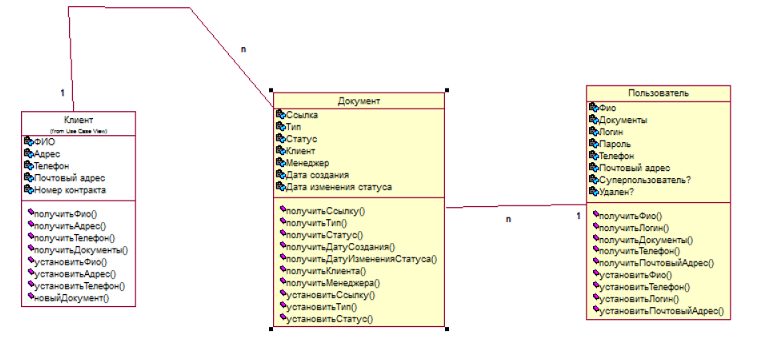
**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** java.io.File;  
**import** java.io.IOException;  
  
*/\*\*  
 \* Created by User on 16.04.2017.  
 \*/***public class** GeneralForm {  
 **static void** showForm() {  
  
 JFrame startFrame = **new** JFrame(**"Главная | Интернет-провайдер"**);  
 startFrame.setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
  
 startFrame.setSize(600,400);  
 startFrame.setLocationRelativeTo(**null**);  
  
 **try** {  
 startFrame.setIconImage(ImageIO.*read*(**new** File(**"out/production/provider/images/diamond\_ico.jpg"**)));  
 } **catch** (IOException ie) {  
 ie.printStackTrace();  
 }  
  
 JPanel mainPanel = **new** JPanel(**new** GridBagLayout());  
 GridBagConstraints c = **new** GridBagConstraints();  
 c.**gridx** = 1;  
 Insets standardInset = **new** Insets(5,0,0,0);  
 c.**insets** = standardInset;  
  
 JButton clientsButton = **new** JButton(**"Клиенты"**);  
 clientsButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** ClientsForm();  
 }  
 });  
  
 JButton myOrdersButton = **new** JButton(**"Мои документы"**);  
 myOrdersButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** OrdersForm(main.*activeManager*.getId());  
 }  
 });  
  
 mainPanel.add(clientsButton,c);  
 mainPanel.add(myOrdersButton,c);  
  
 **if** (main.*activeManager*.getSudo()) {  
 JLabel sudoLabel = **new** JLabel(**"Рут функции:"**);  
  
 JButton managersButton = **new** JButton(**"Менеджеры"**);  
 managersButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** ManagersForm();  
 }  
 });  
  
 JButton ordersStatusesButton = **new** JButton(**"Статусы документов"**);  
 ordersStatusesButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** ListAddForm(**"Статусы документов"**,**"document\_statuses"**);  
 }  
 });  
  
 JButton typesButton = **new** JButton(**"Типы документов"**);  
 typesButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** ListAddForm(**"Типы документов"**,**"document\_types"**);  
 }  
 });  
  
 JButton ordersButton = **new** JButton(**"Все документы"**);  
 ordersButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** OrdersForm(0);  
 }  
 });  
  
 JButton queryButton = **new** JButton(**"Ручные запросы"**);  
 queryButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** QueryForm();  
 }  
 });  
  
 JButton backupButton = **new** JButton(**"Работа с бэкапами"**);  
 backupButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** BackupForm();  
 }  
 });  
  
 c.**insets** = **new** Insets(15,0,5,0);  
 mainPanel.add(sudoLabel,c);  
  
 c.**insets** = standardInset;  
 mainPanel.add(managersButton,c);  
 mainPanel.add(ordersStatusesButton,c);  
 mainPanel.add(typesButton,c);  
 mainPanel.add(ordersButton,c);  
 mainPanel.add(queryButton,c);  
 mainPanel.add(backupButton,c);  
 }  
  
 startFrame.add(mainPanel,BorderLayout.***CENTER***);  
 startFrame.setVisible(**true**);  
 }  
}

**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** javax.swing.table.AbstractTableModel;  
**import** javax.swing.table.DefaultTableModel;  
**import** java.awt.\*;  
**import** java.awt.event.\*;  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.sql.ResultSet;  
**import** java.sql.ResultSetMetaData;  
**import** java.sql.SQLException;  
**import** java.util.ArrayList;  
**import** java.util.Vector;  
  
  
**public class** BackupForm {  
  
 **private** JFrame **startFrame**;  
 **private** JTextField **backupField** = **new** JTextField(**""**,30);  
  
 **public** BackupForm() {  
  
 UIManager.*put*(**"OptionPane.yesButtonText"**, **"Да"**);  
 UIManager.*put*(**"OptionPane.noButtonText"**, **"Нет"**);  
  
 **startFrame** = **new** JFrame(**"Бэкапы | Интернет-провайдер"**);  
 **startFrame**.setDefaultCloseOperation(WindowConstants.***DISPOSE\_ON\_CLOSE***);  
  
 **startFrame**.setSize(600,400);  
 **startFrame**.setLocationRelativeTo(**null**);  
  
 **try** {  
 **startFrame**.setIconImage(ImageIO.*read*(**new** File(**"out/production/provider/images/diamond\_ico.jpg"**)));  
 } **catch** (IOException ie) {  
 ie.printStackTrace();  
 }  
  
 JPanel fieldsPanel = **new** JPanel(**new** GridBagLayout());  
 GridBagConstraints c = **new** GridBagConstraints();  
  
  
 c.**insets** = **new** Insets(5,5,5,5);  
  
 JButton makeBackupBtn = **new** JButton(**"Сделать бэкап"**);  
 makeBackupBtn.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 String res = main.*Backupdbtosql*();  
 **if** (res == **"fail"**)  
 JOptionPane.*showMessageDialog*(**startFrame**,**"Невозможно сделать бэкап."**,**"Ошибка"**,JOptionPane.***ERROR\_MESSAGE***);  
 **else** JOptionPane.*showMessageDialog*(**startFrame**,**"Бэкап сохранен в "** + res,**"Результат"**,JOptionPane.***INFORMATION\_MESSAGE***);  
  
 }  
 });  
  
 c.**gridx** = 0;  
 c.**gridy** = 0;  
 fieldsPanel.add(makeBackupBtn,c);  
  
 c.**gridy** = 1;  
 fieldsPanel.add(**new** JLabel(**"Восстановление бэкапа:"**),c);  
 c.**gridy** = 2;  
 fieldsPanel.add(**new** JLabel(**"Название файла:"**),c);  
 c.**gridx** = 1;  
 fieldsPanel.add(**backupField**,c);  
 c.**gridx** = 0;  
 c.**gridy** = 3;  
 JButton restoreBtn = **new** JButton(**"Восстановить"**);  
 fieldsPanel.add(restoreBtn,c);  
 restoreBtn.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **if** (**backupField**.getText().length() > 0) {  
 String file = **backupField**.getText();  
 main.*Restoredbfromsql*(file);  
 }  
 }  
 });  
  
 **startFrame**.add(fieldsPanel,BorderLayout.***NORTH***);  
  
 **startFrame**.setVisible(**true**);  
 }  
}

**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** javax.swing.table.AbstractTableModel;  
**import** javax.swing.table.DefaultTableModel;  
**import** java.awt.\*;  
**import** java.awt.event.\*;  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.sql.ResultSet;  
**import** java.sql.ResultSetMetaData;  
**import** java.sql.SQLException;  
**import** java.util.ArrayList;  
**import** java.util.Vector;  
  
  
**public class** QueryForm {  
  
 **private** JFrame startFrame;  
 **private** JTable queryTable;  
  
 **private** JTextArea queryField = **new** JTextArea(10,30);  
  
 **public** QueryForm() {  
  
 UIManager.put(**"OptionPane.yesButtonText"**, **"Да"**);  
 UIManager.put(**"OptionPane.noButtonText"**, **"Нет"**);  
  
 startFrame = **new** JFrame(**"Запросы | Интернет-провайдер"**);  
 startFrame.setDefaultCloseOperation(WindowConstants.DISPOSE\_ON\_CLOSE);  
  
 startFrame.setSize(900,400);  
 startFrame.setLocationRelativeTo(**null**);  
  
 **try** {  
 startFrame.setIconImage(ImageIO.read(**new** File(**"out/production/provider/images/diamond\_ico.jpg"**)));  
 } **catch** (IOException ie) {  
 ie.printStackTrace();  
 }  
  
 JPanel fieldsPanel = **new** JPanel(**new** GridBagLayout());  
 GridBagConstraints c = **new** GridBagConstraints();  
  
 JPanel tablePanel = **new** JPanel(**new** BorderLayout());  
  
 c.insets = **new** Insets(5,5,5,5);  
  
 c.gridx = 0;  
 c.gridy = 0;  
 fieldsPanel.add(**new** JLabel(**"Ваш запрос:"**),c);  
 c.gridy = 1;  
 c.gridwidth = 5;  
 fieldsPanel.add(queryField,c);  
 c.gridwidth = 1;  
  
 JButton queryButton = **new** JButton(**"Выполнить"**);  
 c.gridy = 2;  
 fieldsPanel.add(queryButton,c);  
  
 c.gridy = 3;  
 JButton procBtn1 = **new** JButton(**"Количество клиентов"**);  
 procBtn1.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"CALL get\_clients()"**);  
 }  
 });  
 JButton procBtn2 = **new** JButton(**"Количество документов"**);  
 procBtn2.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"CALL get\_documents()"**);  
 }  
 });  
 JButton procBtn3 = **new** JButton(**"Количество пользователей"**);  
 procBtn3.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"CALL get\_users()"**);  
 }  
 });  
 JButton procBtn4 = **new** JButton(**"Количество статусов"**);  
 procBtn4.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"CALL get\_document\_statuses()"**);  
 }  
 });  
 JButton procBtn5 = **new** JButton(**"Количество типов"**);  
 procBtn5.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"CALL get\_document\_types()"**);  
 }  
 });  
  
 fieldsPanel.add(procBtn1,c);  
 c.gridx = 1;  
 fieldsPanel.add(procBtn2,c);  
 c.gridx = 2;  
 fieldsPanel.add(procBtn3,c);  
 c.gridx = 3;  
 fieldsPanel.add(procBtn4,c);  
 c.gridx = 4;  
 fieldsPanel.add(procBtn5,c);  
 c.gridx = 5;  
  
 c.gridx = 0;  
 c.gridy = 4;  
  
 JButton fncBtn1 = **new** JButton(**"Сложение"**);  
 fncBtn1.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"SELECT plus(5,4) as result"**);  
 }  
 });  
 JButton fncBtn2 = **new** JButton(**"Вычитание"**);  
 fncBtn2.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"SELECT minus(5,4) as result"**);  
 }  
 });  
 JButton fncBtn3 = **new** JButton(**"Умножение"**);  
 fncBtn3.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"SELECT multiple(5,4) as result"**);  
 }  
 });  
 JButton fncBtn4 = **new** JButton(**"Деление"**);  
 fncBtn4.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"SELECT divide(5,4) as result"**);  
 }  
 });  
 JButton fncBtn5 = **new** JButton(**"Остаток"**);  
 fncBtn5.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 queryField.setText(**"SELECT moding(5,4) as result"**);  
 }  
 });  
  
 fieldsPanel.add(fncBtn1,c);  
 c.gridx = 1;  
 fieldsPanel.add(fncBtn2,c);  
 c.gridx = 2;  
 fieldsPanel.add(fncBtn3,c);  
 c.gridx = 3;  
 fieldsPanel.add(fncBtn4,c);  
 c.gridx = 4;  
 fieldsPanel.add(fncBtn5,c);  
 c.gridx = 5;  
   
 queryButton.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 String query = queryField.getText();  
 **if** (query.length() > 0) {  
 **if** (query.substring(0,6).equals(**"SELECT"**) || query.substring(0,4).equals(**"CALL"**)) {  
 ResultSet rs = **null**;  
 **try** {  
 rs = main.db.query(query, **new** String[]{});  
 ResultSetMetaData metaData = rs.getMetaData();  
  
 Vector<String> columnNames = **new** Vector<String>();  
 **int** columnCount = metaData.getColumnCount();  
 **for** (**int** column = 1; column <= columnCount; column++) {  
 columnNames.add(metaData.getColumnName(column));  
 }  
  
 Vector<Vector<Object>> data = **new** Vector<Vector<Object>>();  
 **while** (rs.next()) {  
 Vector<Object> vector = **new** Vector<Object>();  
 **for** (**int** columnIndex = 1; columnIndex <= columnCount; columnIndex++) {  
 vector.add(rs.getObject(columnIndex));  
 }  
 data.add(vector);  
 }  
  
 DefaultTableModel model = **new** DefaultTableModel(data, columnNames);  
 queryTable.setModel(model);  
 model.fireTableDataChanged();  
  
 startFrame.pack();  
 } **catch** (Exception ex) {  
 String message = **""**;  
 **if** (ex.getMessage() != **null**)  
 message = ex.getMessage();  
 **else** message = **"Ошибка в запросе."**;  
 JOptionPane.*showMessageDialog*(**startFrame**,message,**"Ошибка"**,JOptionPane.***ERROR\_MESSAGE***);  
 } **finally** {  
 main.*db*.closeStatementSet(rs);  
 }  
 } **else** {  
 **boolean** go = **true**;  
 **int** result = 0;  
 **try** {  
 result = main.*db*.query(query);  
 } **catch** (SQLException se) {  
 JOptionPane.*showMessageDialog*(**startFrame**,se.getMessage(),**"Ошибка"**,JOptionPane.***ERROR\_MESSAGE***);  
 go = **false**;  
 }  
 **if** (go && result > 0)  
 JOptionPane.*showMessageDialog*(**startFrame**,**"Запрос выполнен"**,**"Результат запроса"**,JOptionPane.***INFORMATION\_MESSAGE***);  
 }  
 }  
 }  
 });  
  
  
 **queryTable** = **new** JTable();  
 JScrollPane ordersScroll = **new** JScrollPane(**queryTable**);  
  
 tablePanel.add(ordersScroll,BorderLayout.***NORTH***);  
  
 startFrame.add(fieldsPanel,BorderLayout.NORTH);  
 startFrame.add(tablePanel,BorderLayout.SOUTH);  
  
 startFrame.setVisible(**true**);  
 }  
}

**ПРИЛОЖЕНИЕ B.**

Диаграмма классов



**ПРИЛОЖЕНИЕ C.**

Диаграмма вариантов использования

