**Введение**

Автоматизированные информационные системы нашли широкое применение практически во всех сферах человеческой деятельности. Создание таких систем позволяет упростить обработку данных и сократить время поиска информации.

Автоматизированные информационные системы активно внедряются в различные учреждения, сети магазинов, предприятий.

Автоматизированные информационные системы нашли своё место и в ведении расчета и учета оплаты ЖКХ. Они позволяют автоматизировать просмотр оплаченных, неоплаченных счетов и формирование квитанций по оплате услуг ЖКХ.

Область использования АИС охватывает как отдельные организации (УК, ТСЖ, ЖСК), так и позволяет работать в едином информационном пространстве (с набором БД или единой БД) различным организациям, оказывающим услуги в сфере ЖКХ.

АИС способны повысить производительность работы персонала, улучшить качество обслуживания клиентов, снизить трудоемкость и напряженность труда персонала, снизить количество ошибок в действиях персонала.

Основной причиной создания и поддерживания АИС является необходимость ведения учёта информации об оплаченных и неоплаченных счетах ЖКХ.

Целью является разработка автоматизированной информационной системы для учета оплаченных и неоплаченных счетов ЖКХ, и дальнейшего вывода отчета в MS Word.

Для реализации цели необходимо решить следующие задачи:

* проанализировать предметную область;
* сформировать требования к системной разработке;
* произвести выбор инструментальных и программных средств;
* реализовать автоматизированную информационную систему;
* протестировать автоматизированную информационную систему;
* разработать техническую и эксплуатационную документацию.

1. **Разработка системного проекта**

**1.1 Назначение разработки**

Автоматизированная информационная система по ведению базы данных расчета и уплаты ЖКХ предназначена для автоматизации формирования квитанций и учета оплаты ЖКХ.

* 1. **Требования к функциональным характеристикам**

Автоматизированная информационная система расчета и учета оплаты ЖКХ должна обеспечивать выполнение следующих функций:

* авторизация (главный администратор, диспетчер);
* ведение и редактирование данных (добавление, удаление, редактирование);
* просмотр текущего состояния об оплате счета;
* изменение статуса об оплате счета;
* формирование квитанции за месяц;
* расчет стоимости оплаты за месяц;
* составление отчёта учета оплаченных квитанций за месяц;
* поиск записей с помощью фильтра (по лицевому счету).

Входная информация:

* ввод номера лицевого счета.

Выходная информация:

* формирование отчета по оплате счетов ЖКХ;
* вывод квитанции на печать в файле Word.

**1.3 Требования к надежности и безопасности**

Разрабатываемое ПО должно иметь:

* контроль ввода данных;
* резервное копирование БД;
* разграничение прав доступа.

**1.4 Требования к составу и параметрам технических средств**

Минимальные системные требования:

* тактовая частота процессора - 2000 Гц;
* объем оперативной памяти 2 Гб;
* объем свободного дискового пространства 25 МБ;
* разрешение монитора 1366х768;
* наличие устройства чтения компакт-дисков.

Состав программных и аппаратных средств:

* Microsoft Office Access;
* Microsoft Office Word.

**1.5 Требование к информационной и программной совместимости**

Программа должна работать в операционных системах Windows. Все формируемые квитанции должны иметь возможность экспорта в программу для работы с электронными таблицами Microsoft Office Word 2007/2019.

1. **Разработка технического проекта**
   1. **Анализ требований и определение спецификаций ПО**
      1. **Выбор технологии проектирования**

При проектировании программного продукта применяются 2 подхода: объектно-ориентированный и структурный:

Объектно-ориентированное проектирование (ООП) предназначено организовывать программные системы с БД. Через ООП можно "скрыть" внутреннюю структуру базы от конечного пользователя. Если БД будут менять, то не потребуется полное переписывание всего проекта.

С помощью ООП можно разбить сложную логику на несколько простых классов. Преимущество в том, что можно создавать универсальный интерфейс и узкоспециализированные классы.

При написании АИС главное преимущество ООП системы состоит в том, что оно упрощает задачу внесения изменений, поскольку представление состояния объекта не указывает на неё влияние.

Для проектирования приложения выбран структурный подход для построения ER-диаграммы и объектно-ориентированный подход для построения диаграммы вариантов использования, и диаграммы деятельности.

Основой взаимосвязи является общность ряда категорий и понятий обоих подходов (про­цесс и вариант использования, сущность и класс и др.). Эта взаи­мосвязь может проявляться в различных формах. Так, одним из возможных вариантов является использование структурного анализа как основы для объектно-ориентированного проектирова­ния. При этом структурный анализ следует прекращать, как только структурные модели начнут отражать не только деятельность организации (бизнес-процессы), а и систему ПО. После выполнения структурного анализа можно различными способами приступить к определению классов и объектов. Так, если взять какую-либо отдельную диаграмму потоков данных, то кандидатами в классы могут быть элементы структур данных.

Другой формой проявления взаимосвязи можно считать интеграцию объектной и реляционной технологий. Реляционные СУБД являются на сегодняшний день основным средством реализации крупномасштабных баз данных и хранилищ данных. Причины этого достаточно очевидны: реляционная технология используется достаточно долго, освоена огромным количеством пользователей и разработчиков, стала промышленным стандартом, в нее вложе­ны значительные средства и создано множество корпоративных БД в самых различных отраслях, реляционная модель проста и имеет строгое математическое основание; существует большое разнообразие промышленных средств проектирования, реализации и эксплуатации реляционных БД. Вследствие этого реляционные БД в основном используются для хранения и поиска объектов в так называемых объектно-реляционных системах.

Объектно-ориентированное проектирование имеет точки соприкосновения с реляционным проектированием. Например, как было отмечено выше, классы в объектной модели могут некоторым образом соответствовать сущностям (в качестве упражнения можно предложить детально проанализировать все сходства и различия диаграмм «сущность-связь» и диаграмм классов). Как правило, такое соответствие имеет место только на ранней стадии разработки системы. В дальнейшем, разумеется, цели объектно-ориентированного проектирования (адекватное моделирование предметной области в терминах взаимодействия объектов) и разработки реляционной БД (нормализация данных) расходятся. Таким образом, единственно возможным средством пре­одоления данного разрыва является построение отображения между объектно-ориентированной и реляционной технологиями, которое в основном сводится к отображению между диаграммами классов и реляционной моделью. Все это уменьшает риск создания сложных систем ПО, так как она предполагает эволюционный путь развития системы на базе относительно небольших подсистем.

* + 1. **Построение моделей**

При объектно-ориентированном проектировании были разработаны диаграмма вариантов использования, диаграмма деятельности, ER-диаграмма.

Диаграмма вариантов использования позволяет наглядно представить ожидаемое поведение программной системы. Основные понятия диаграммы вариантов использования являются: действующее лицо, вариант использования и связь.

Условные обозначения диаграммы вариантов использования:

Актёр – действующее лицо, которое может взаимодействовать с системой. Он выступает как роль, которая определенно относится к программной системе.

Актёр представлен в соответствии с рисунком 2.1.2.1.



Рисунок 2.1.2.1 – Актёр

Вариант использования - внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами. Вариант использования представлен в соответствии с рисунком 2.1.2.2.

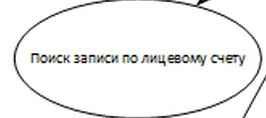


Рисунок 2.1.2.2 – Вариант использования

Связь – взаимодействие действующих лиц или соответствующих вариантов использования. Представлено в соответствии с рисунком 2.1.2.3.

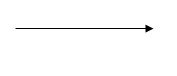


Рисунок 2.1.2.3 – Связь

Диаграмма вариантов использования представлена в соответствии с рисунком 2.1.2.4.

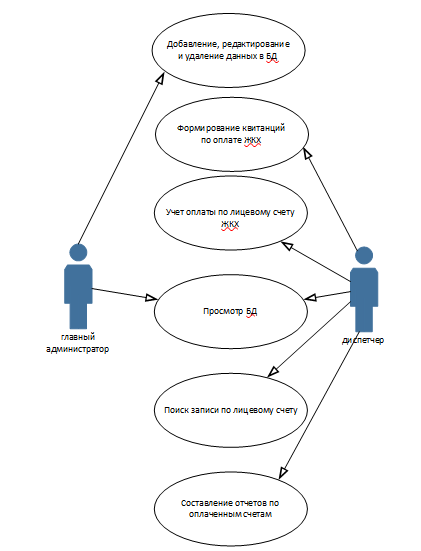
****

Рисунок 2.1.2.4 – Диаграмма вариантов использования

Сценарий диаграммы использования, раскрывает логическую последовательность отдельных действий, которые исполняют актеры. Сценарий диаграммы вариантов использования представлен в соответствии с таблицей 2.1.

Таблица А.1 – Сценарий диаграммы вариантов использования

|  |  |
| --- | --- |
| Вариант использования | Формирование квитанции по оплате ЖКХ |
| Актеры | Главный администратор, диспетчер |
| Краткое описание | Главный администратор – может удалять/редактировать базы данных, производить поиск информации, просматривать базу данных.  Диспетчер может формировать отчеты об оплате квитанций за месяц, выводить квитанции на печать |
| Цель | Учет оплаты ЖКХ по лицевому счету |
| Тип | Базовый |
| Ссылки на другие варианты использования | Добавление, редактирование и удаление данных в БД, учет оплаты по лицевому счету ЖКХ, просмотр БД, поиск записи по лицевому счету, составление отчетов по оплаченным счетам |

Диаграмма деятельности - диаграмма, на которой показаны действия, состояния которых описаны на [диаграмме состояний](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B9_(UML)). Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Условные обозначения диаграммы деятельности:

Под началом понимается включение программы, представлено в соответствии с рисунком 2.1.2.5.



Рисунок 2.1.2.5 – Начало

Под условие понимается выбор какого-либо действия, представлено в соответствии с рисунком 2.1.2.6.



Рисунок 2.1.2.6 – Условие

Под действием понимается выполнение какой-либо операции программой, представлено в соответствии с рисунком 2.1.2.7.

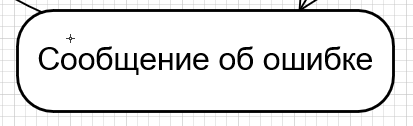


Рисунок 2.1.2.7 – Действие

Под связью понимается соединение определенных операций, представлено в соответствии с рисунком 2.1.2.8.



Рисунок 2.1.2.8 – Связь

Под разделением связей понимается выбор одного из нескольких действий, представлено в соответствии с рисунком 2.1.2.9.

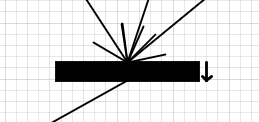


Рисунок 2.1.2.9 – Разделение связей

Под концом понимается завершение программы, либо выход из неё, представлено в соответствии с рисунком 2.1.2.10.



Рисунок 2.1.2.10 – Конец

Диаграмма деятельности представлена в соответствии с рисунком 2.1.2.11.



Рисунок 2.1.2.11 – Диаграмма деятельности

ER – диаграмма обеспечивает стандартный способ определения данных и отношений между ними. Она включает сущности и взаимосвязи предметной области.

Основные понятия ER – диаграмм.

Сущность – это класс однотипных объектов, информация о которых должна быть учтена в модели. Представлено в соответствии с рисунком 2.1.2.11.

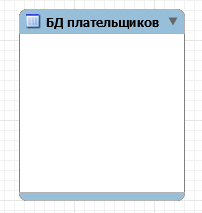


Рисунок 2.1.2.11 – Сущность

Атрибут сущности – это именованная характеристика, являющаяся некоторым свойством сущности, представлен в соответствии с рисунком 2.1.2.12.

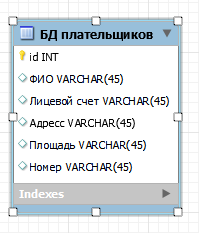


Рисунок 2.1.2.12 – Сущность с атрибутами

Ключ сущности – это не избыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Ключевые атрибуты изображаются подчёркиванием, представлено в соответствии с рисунком 2.1.2.13.

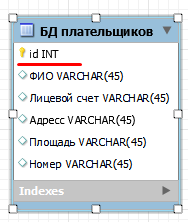


Рисунок 2.1.2.13 – Сущность с ключом

Связь – это некоторая ассоциация между двумя сущностями. Представлено в соответствии с рисунком 2.1.2.14.

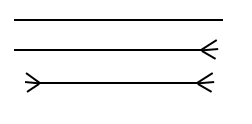


Рисунок 2.1.2.14 – Связи

ER – диаграмма представлена в соответствии с рисунком 2.1.2.15.

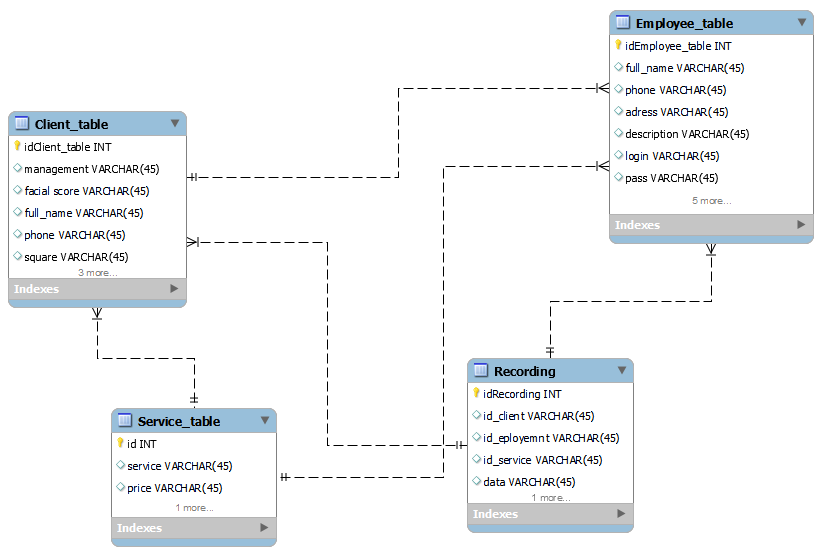


Рисунок 2.1.2.15 - ER – диаграмма

* 1. **Детальное проектирование ПО**

Прототипы могут быть горизонтальные и вертикальные, одноразовые и эволюционные, бумажные и электронные.

Горизонтальный прототип моделирует интерфейс пользователя приложения, не затрагивая логики обработки и структур данных.

Вертикальный прототип направлен не столько на проектирование интерфейса пользователя, сколько на реализацию вертикального «среза» системы, затрагивая все уровни ее реализации.

Одноразовый прототип создается, когда нужно быстро получить маркер разрабатываемой программной системы, те или иные ее аспекты и компоненты.

Эволюционный прототип создается как первое приближение системы, призванное стать впоследствии самой системой.

Бумажный прототип наброски интерфейсов на бумаге.

При проектировании ПП продукта был выбран горизонтальный прототип, так как его используют, когда необходимо прояснить нечёткие требования, имеющие многоальтернативную реализацию. Для горизонтальных прототипов проще всего будет использовать программную среду реализации, в которой будет разрабатываться конечный вариант системы.

Прототип авторизации пользователя представлен в соответствии с рисунком 2.1.1.

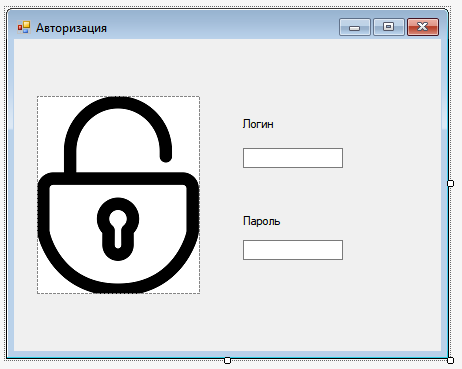


Рисунок 2.1.1 – Прототип авторизации пользователя

Горизонтальный прототип главного окна представлен в соответствии с рисунком 2.1.2.

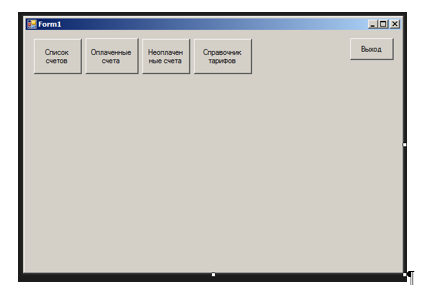


Рисунок Б.2 – Прототип главного окна

Горизонтальный прототип окна списка счетов представлено в соответствии с рисунком 2.1.3.

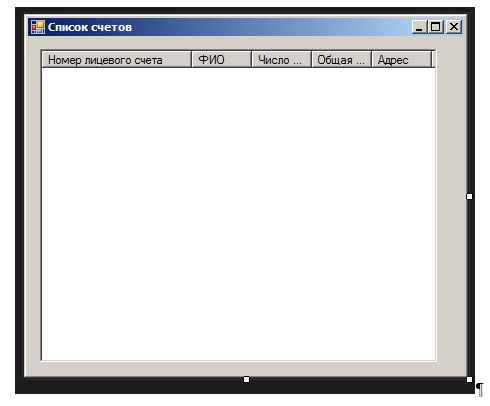


Рисунок 2.1.3. – Прототип окна информации о собственнике

Горизонтальный прототип окна просмотра справочника тарифов представлен в соответствии с рисунком 2.1.4.

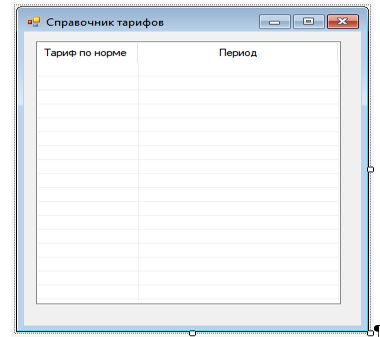


Рисунок 2.1.4 – Прототип окна просмотра справочника тарифов

Горизонтальный прототип окна формирования отчета представлен в соответствии с рисунком 2.1.5.

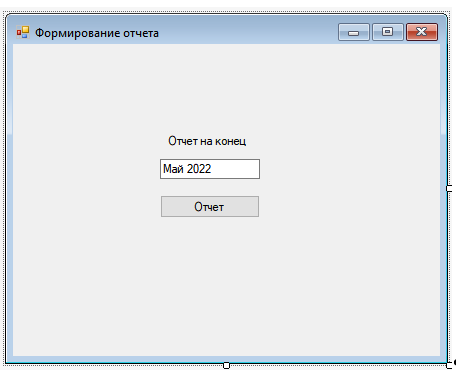


Рисунок 2.1.5– Прототип окна формирования отчета

Горизонтальный прототип окна редактирования БД в режиме главного администратора представлен в соответствии с рисунком 2.1.6.

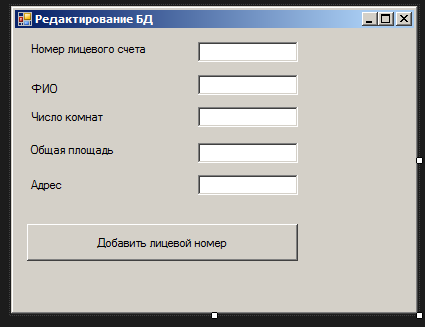


Рисунок 2.1.6– Прототип окна редактирования БД

Горизонтальный прототип окна оплаченных счетов в режиме диспетчера представлен в соответствии с рисунком 2.1.7.

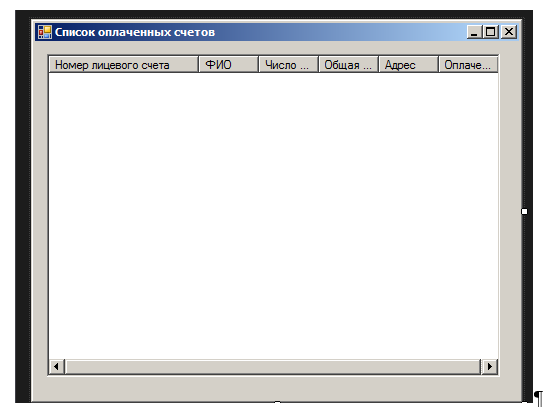


Рисунок 2.1.7– Прототип окна оплаченных счетов в режиме диспетчера

Горизонтальный прототип окна неоплаченных счетов в режиме диспетчера представлен в соответствии с рисунком 2.1.8.

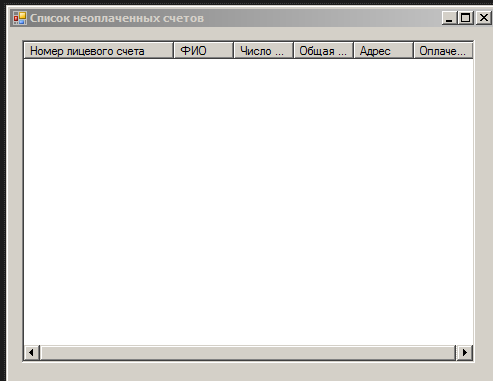


Рисунок 2.1.8– Прототип окна неоплаченных счетов в режиме диспетчера

Прототип вывода квитанции на печать об неоплаченных счетах в MS Office Word представлен в соответствии с рисунком 2.1.9.

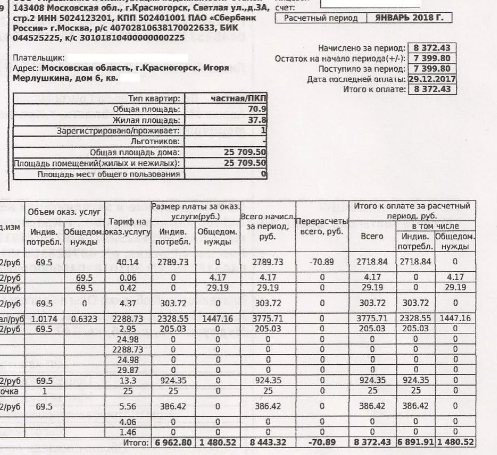


Рисунок 2.1.9 – Прототип вывода квитанции на печать об неоплаченных счетах в MS Office Word

1. **Реализация**
   1. **Обоснование выбора средств разработки**

При проектировании приложения было выбрано: операционная система, среда разработки, язык программирования и база данных.

К основным ОС относятся: MS Windows, Linux, MacOS.

MS Windows – семейство [коммерческих](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D1%80%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)(OC) корпорации [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), ориентированных на управление с помощью [графического интерфейса](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F).

Плюсы ОС Windows:

* система достаточно проста в использовании;
* большинство программ как платных, так и бесплатных поддерживают именно эту операционную систему;
* поддержка различного оборудования (веб-камеры, принтеры и т.д.);
* стабильность работы.

Минусы ОС Windows:

* платная ОС;
* большинство вирусов написано под эту ОС;
* довольно частые сбои.

MacOS — операционная система от компании Apple. Установлена как основная операционная система на продуктах компании.

Плюсы ОС MacOS:

* удобный пользовательский интерфейс;
* стоимость входит в стоимость оборудования;
* высокая производительность;
* низкая возможность заражения вирусами.

Минусы ОС MacOS:

* дорогое программное обеспечение;
* нельзя установить на оборудование другой компании;

Linux — это **операционная система, ядро которой распространяется на бесплатной основе.** Она состоит из ядра системы и набора небольших программ, взаимодействующих с этим ядром.

Плюсы ОС Linux:

* бесплатное свободно-распространяемое ПО;
* практически нет вирусов;
* не зависит от разработчика.

Минусы ОС Linux:

* поддержка небольшого количества оборудования;
* небольшое количество прикладных программ.

Для разработки была выбрана ОС Windows, т.к. система достаточно проста в использовании, поддерживает большое количество программ, как платных, так и бесплатны и обеспечивает стабильность работы.

К основным IDE относятся Visual Studio, JetBrains Rider.

Visual Studio – это интегрированная среда разработки программного обеспечения от компании Microsoft. С помощью Visual Studio можно создавать приложения для Windows, iOS, Android и других платформ.

Плюсы:

* встроенный Web-сервер;
* поддержка множества языков при разработке;
* меньше кода для написания;
* интуитивный стиль кодирования;
* возможности отладки;
* поддержка Microsoft.

NetBeans – бесплатная среда разработки с открытым исходным кодом. Подходит для редактирования существующих проектов или создания новых. NetBeans предлагает простой интерфейс, который поставляется с большим количеством удобных шаблонов проектов.

Плюсы:

* интуитивный и понятный интерфейс;
* возможность подключения библиотек;
* возможность осуществлять поддерживать ПО весь цикл разработки;
* совместимость с платформами Windows, Linux, OS X.

В качестве среды программирования была выбрана Visual Studio, потому что присутствует поддержка Microsoft, русифицирование и поддержка графического редактора для создания проекта.

К основным языкам программирования относятся С#, Python, C++.

C# — объектно-ориентированный язык программирования, который имеет статическую типизацию, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, комментарии в формате XML и быстрое подключение базы данных.

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. В то же время стандартная библиотека включает большой объём полезных функций.

C++ — компилируемый, статически типизированный язык программирования общего назначения. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

В качестве языка программирования был выбран язык программирования C#, т.к. это высокоуровневый язык программирования, он удобен в использовании и поддерживает графический интерфейс.

К основным базам данных относятся SQLite, MySQL и Microsoft Access.

SQLite - библиотека, встраиваемая в приложение, которое её использует. Будучи файловой БД, она предоставляет отличный набор инструментов для более простой обработки любых видов данных. Когда приложение использует SQLite, их связь производится с помощью функциональных и прямых вызовов файлов, содержащих данные, а не какого-то интерфейса, что повышает скорость и производительность операций.

Плюсы:

* файловая;
* стандартизированная;
* отлично подходит для разработки и даже тестирования.

MySQL — это самая популярная из всех крупных серверных БД. MySQL - это СУБД, которая поддерживает SQL. SQL — это универсальный язык, который поддерживают все системы управлением базами данных, поэтому, зная его, вы сможете работать с любыми СУБД. MySQL предлагает широкий функционал. Приложения общаются с базой данных через процесс-демон.

Плюсы:

* легко устанавливается;
* поддерживает большую часть функционала SQL;
* безопасность;
* может работать с действительно большими объёмами данных, и неплохо походит для масштабируемых приложений.

Microsoft Access – система управления базами данных, имеющая широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самом Access можно писать приложения, работающие с базами данных.

Плюсы:

* простой графический интерфейс, который позволяет не только создавать собственную базу данных, но и разрабатывать приложения, используя встроенные средства;
* хранит все данные в одном файле, хотя и распределяет их по разным таблицам, как и положено реляционной СУБД. К этим данным относится не только информация в таблицах, но и другие объекты базы данных;
* распространенность, которая обусловлена тем, что Access является продуктом компании Microsoft.

В качестве базы данных была выбрана Access, т.к она удобна в использовании, может работать с большим количеством информации и проста в использовании. Имеет широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных.

**3.2 Описание основных программных модулей**

Основные программные модули представлены в таблице 3.2.1.

Таблица 3.2.1 – Основные программные модули

|  |  |
| --- | --- |
| **Название модуля** | **Описание** |
| 1 | 2 |
| Authorization.cs | Форма авторизации |
| public Authorization() | Инициализация формы авторизации |
| private void button2\_Click(object sender, EventArgs e)  {  Application.Exit();  } | Кнопка выхода из программы |
| private void button1\_Click(object sender, EventArgs e)  { | Кнопка для авторизации в учетной записи |
| Backup.cs | Форма резервного копирования |
| public Backup() | Инициализация формы резервного копирования |

Продолжение таблицы 3.2.1

|  |  |
| --- | --- |
| private void button2\_Click(object sender, EventArgs e) | Кнопка для подтверждения резервного копирования |
| private void button1\_Click(object sender, EventArgs e) | Кнопка возвращения обратно в меню |
| private void button5\_Click(object sender, EventArgs e) | Кнопка возвращения обратно в меню |
| Client\_table.cs | Форма добавления плательщика |
| private void button2\_Click(object sender, EventArgs e) | Кнопка добавления плательщика |
| private void button3\_Click(object sender, EventArgs e) | Кнопка редактирования данных плательщика |
| private void button4\_Click(object sender, EventArgs e) | Кнопка удаления плательщика |
| Edit.cs | Форма редактирования статуса счета |
| private void button4\_Click(object sender, EventArgs e) | Кнопка изменения статуса счета |
| private void button1\_Click(object sender, EventArgs e) | Кнопка добавления статуса счета |
| private void button3\_Click(object sender, EventArgs e) | Кнопка удаления статуса счета |

Продолжение таблицы 3.2.1

|  |  |
| --- | --- |
| private void Fam\_KeyPress(object sender, KeyPressEventArgs e) | Контроль ввода данных |
| private void Ph\_KeyPress(object sender, KeyPressEventArgs e) |
| private void Mod\_KeyPress(object sender, KeyPressEventArgs e) |
| private void Square\_KeyPress(object sender, KeyPressEventArgs e) |
| private void Facial\_Score\_KeyPress(object sender, KeyPressEventArgs e) |  |
| Employ.cs | Форма просмотра пользователей программы |
| private void button2\_Click(object sender, EventArgs e) | Кнопка добавления пользователя из БД программы |
| private void button3\_Click(object sender, EventArgs e) | Кнопка редактирования пользователя из БД программы |
| private void button4\_Click(object sender, EventArgs e) | Кнопка удаления пользователя из БД программы |
| Menu.cs | Форма меню |
| private void button4\_Click(object sender, EventArgs e) | Кнопка перехода на форму таблицы плательщиков |
| private void button5\_Click(object sender, EventArgs e) | Кнопка перехода на форму редактирования счета |
| private void button2\_Click(object sender, EventArgs e) | Кнопка перехода на форму платежных документов |
| private void button6\_Click(object sender, EventArgs e) | Кнопка перехода на форму редактирования пользователей |
| private void button7\_Click(object sender, EventArgs e) | Кнопка перехода на форму тарифов ЖКХ |
| Receipt.cs | Форма просмотра тарифов ЖКХ |

Продолжение таблицы 3.2.1

|  |  |
| --- | --- |
| private void button1\_Click(object sender, EventArgs e) | Кнопка для оформления платежного документа |
| Service.cs | Кнопка удаления пропуска |
| private void button1\_Click(object sender, EventArgs e) | Кнопка добавления услуги ЖКХ |
| private void button3\_Click(object sender, EventArgs e) | Кнопка изменения услуги ЖКХ |
| private void button4\_Click(object sender, EventArgs e) | Кнопка удаления услуги ЖКХ |

**4 Тестирование программного продукта**

Тестирование программного обеспечения – процесс исследования, испытания [программного продукта](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Существует несколько признаков, по которым принято производить классификацию видов тестирования. Обычно выделяют следующие:

* тестирование переходов между состояниями – тестирование состояния программы и меню программы;
* условия гонок – вмешательство в работу программы, когда она выполняет переход между состояниями, обработку данных, ввод-вывод данных;
* нагрузочные испытания – один из видов тестирования граничных условий. Сначала запускают программу в нормальных условиях, а затем в условиях, для которой она не предназначена;
* тестирование функциональной эквивалентности – при тестировании функциональной эквивалентности сравниваются результаты вычислений разными программами одной и той же математической функции.

Тестирование программного продукта представлено в таблице 4.1

Таблица 4.1- Тестирование

|  |  |  |
| --- | --- | --- |
| **№** | **Испытание** | **Результат** |
| **Форма «Authorization»** | | |
| **1** | Ввод логина и пароля и нажатие на кнопку «Войти» | При правильном вводе пароля и логина происходит переход на главную форму при неправильном сообщение о вводе неверных данных. |
| **Форма «Menu»** | | |
| **2** | Нажатие на кнопку «Таблица плательщиков» | Закрытие формы главного меню и открытие формы просмотра таблицы плательщиков |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **3** | | Нажатие на кнопку «Таблица пользователей» | Закрытие формы главного меню и открытие формы просмотра таблицы пользователей | |
| **4** | | Нажатие на кнопку «Редактирование счета» | Закрытие формы главного меню и открытие формы редактирования счета | |
| **5** | | Нажатие на кнопку «Тарифы ЖКХ» | Закрытие формы главного меню и открытие формы просмотра тарифов ЖКХ | |
| **6** | | Нажатие на кнопку «Платежные документы» | Закрытие формы главного меню и открытие формы просмотра платежных документов | |
| **7** | | Нажатие на кнопку «Резервное копирование БД» | Закрытие формы главного меню и открытие формы просмотра резервного копирования БД.  Выводится форма с выбором конкретного файла для резервного копирования. После выбора делается резервная копия БД. | |
| **8** | | Нажатие на кнопку «Выход» | После нажатия форма главного меню закрывается, происходит выход из учетной записи. | |
| **Форма «Client\_table»** | | | | |
| **9** | Нажатие на кнопку «Добавить плательщика» | | | Возможность добавления плательщика на этой же форме |
| **10** | Нажатие на кнопку «Изменить плательщика» | | | Возможность редактирования плательщика на этой же форме |
| **11** | Нажатие на кнопку «Удалить плательщика» | | | Возможность удаления плательщика на этой же форме |
| **Форма «Edit»** | | | | |
| **12** | Нажатие на кнопку «Изменить статус» | | | Возможность изменения статуса оплаты на этой же форме с помощью выпадающего списка |
| **13** | Нажатие на кнопку «Добавить счет» | | | Возможность добавления нового счета для оплаты услуг |
| **14** | Нажатие на кнопку «Удалить» | | | Возможность удаления счета |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| **Форма «Employ»** | | |
| **15** | Нажатие на кнопку «Добавить пользователя» | Возможность добавления пользователя программы на этой же форме |
| **16** | Нажатие на кнопку «Редактировать пользователя» | Возможность редактирования пользователя программы на этой же форме |
| **17** | Нажатие на кнопку «Удалить пользователя» | Возможность удаления пользователя программы на этой же форме |
| **Форма «Receipt»** | | |
| **18** | Нажатие кнопку «Печать квитанции» | Открытие файла Word, занесение выбранного плательщика и данных об оплате |
| **Форма «Reports»** | | |
| **19** | Нажатие на кнопку «Печать» под заголовком «Отчет неоплаченных счетов | Открытие файла Word, занесение информации об неоплаченных счетах за период |
| **20** | Нажатие на кнопку «Печать» под заголовком «Отчет оплаченных счетов | Открытие файла Word, занесение информации об оплаченных счетах за период |
| **Форма «Service»** | | |
| **19** | Нажатие на кнопку «Добавить услугу» | Возможность добавления услуги ЖКХ на этой же форме |
| **20** | Нажатие на кнопку «Изменить услугу» | Возможность редактирования услуги ЖКХ на этой же форме |
| **21** | Нажатие на кнопку «Удалить услугу» | Возможность удаления услуги ЖКХ на этой же форме |

**Заключение**

В ходе выполнение курсового проекта была разработана автоматизированная информационная система по ведению базы данных расчета и учета оплаты ЖКХ. Предназначена для минимизации трудовых затрат работников при обработке данных обработки данных, за счет облегчения формирования квитанции, автоматизированного добавления данных через форму и учет оплаченных и неоплаченных счетов.

Автоматизированная информационная система выполняет:

* просмотр личных данных: плательщиков, пользователей;
* изменение статуса счета плательщика;
* печать платежного договора оплаты услуг ЖКХ;
* печать отчета оплаченных и неоплаченных счетов;
* возможность ввода нового тарифа ЖКХ.

Программный продукт содержит все заявленные требования, которые были поставлены на этапе постановки задачи:

* авторизация администратора и диспетчера;
* добавление, удаление, редактирование и резервное копирование базы данных администратором;
* поиск информации с использованием фильтра: по лицевому счету
* возможность просмотра информации;
* расчет стоимости оплаты ЖКХ за месяц;
* иметь возможность резервного копирования БД;
* иметь возможность динамического поиска записей с помощью фильтра (по лицевому счету);
* иметь возможность формирования квитанции;

В дальнейшем возможна доработка программы, добавление новых функций по ведению базы данных.

**Литература**

1. Рудаков А.В. "Технология разработки программного обеспечения" -М.:Академия, 2018 – 206 c.
2. Агуров, Павел C#. Сборник рецептов / Павел Агуров. - М.: "БХВ-Петербург", 2018 - 432 c.
3. Бишоп, Дж. C# в кратком изложении / Дж. Бишоп, Н. Хорспул. - М.: Бином. Лаборатория знаний, 2019 - 472 c.
4. Марченко, А. Л. Основы программирования на С# 2.0 / А.Л. Марченко. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2021 - 552 c.
5. Смоленцев, Н. К. MATLAB. Программирование на Visual С#, Borland JBuilder, VBA (+ CD-ROM) / Н.К. Смоленцев. - М.: ДМК Пресс, 2011. - 456 c.