**АННОТАЦИЯ**

Отчет о курсовой работе: 46 с., 9 рис., 2 приложения, 7 источников.

Объект исследования – бюро находок.

Предмет исследования – информационная система бюро находок.

Цель работы – разработка информационной системы «Бюро находок и забытых вещей» с возможностью её дальнейшей эксплуатации.

Метод исследования – анализ возможностей языка С# для создания и обработки входных и выходных данных .

В работе были использованы возможности программных продуктов Visual Studio.

В результате решения задачи было разработана информационная система «Бюро находок и забытых вещей». Приложение позволяет задавать необходимую информацию о потерянных/найденных вещах, о человеке, на который их нашел: ФИО, контактная информация (e-mail и телефон).

Информационная система «Бюро находок и забытых вещей» может использоваться как работниками этого же бюро, либообычными пользователями для поиска утраченных вещей.

С#, ПРОГРАММНЫЙ ПРОДУКТ, ИНФОРМАЦИОННАЯ СИСТЕМА, БЮРО НАХОДОК И ЗАБЫТЫХ ВЕЩЕЙ , VISUAL STUDIO, WPF

**СОДЕРЖАНИЕ**

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc40116426)

[1.1 Состояние вопроса 7](#_Toc40116427)

[1.2 Актуальность и цель работы 9](#_Toc40116428)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 12](#_Toc40116429)

[2.1 Описание области применения и исходных данных приложения 12](#_Toc40116430)

[2.2 Требования к пользовательским интерфейсам 12](#_Toc40116431)

[2.3 Требования к аппаратным, программным и коммуникационным интерфейсам 17](#_Toc40116432)

[2.4 Требования к пользователям продукта 18](#_Toc40116433)

[2.5 Требования к адаптации на месте 18](#_Toc40116434)

[2.6 Функции продукта 18](#_Toc40116435)

[2.7 Ограничения 19](#_Toc40116436)

[3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ 20](#_Toc40116437)

[3.1 Интегрированная среда разработки Visual Studio 20](#_Toc40116438)

[3.2 Язык программирования С# 22](#_Toc40116439)

[3.3 Система для построения клиентских приложений Windows WPF 24](#_Toc40116440)

[4 РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ «БЮРО НАХОДОК И ПОТЕРЯННЫХ ВЕЩЕЙ» 26](#_Toc40116441)

[4.1 Входные и выходные данные приложения 26](#_Toc40116442)

[4.2 Проектирование структуры приложения 26](#_Toc40116443)

[4.3 Описание объектов и их взаимодействия для ООП 27](#_Toc40116444)

[5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 31](#_Toc40116445)

[5.1 Аппаратные и программные средства создания и эксплуатации интернет-приложения 31](#_Toc40116446)

[5.2 Описание контрольных примеров 32](#_Toc40116447)

[5.3 Руководство пользователя 33](#_Toc40116448)

[ЗАКЛЮЧЕНИЕ 35](#_Toc40116449)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc40116450)

[ПРИЛОЖЕНИЕ А Экранные формы 37](#_Toc40116451)

[ПРИЛОЖЕНИЕ Б Фрагменты листинга 42](#_Toc40116452)

**ВВЕДЕНИЕ**

Бюро находок — организация, занимающаяся каталогизацией, хранением и возвратом утерянных вещей.

Обычно бюро находок представляют собой муниципальную организацию. На некоторых больших предприятиях, а также в общественных местах с большим потоком посетителей (вокзалы, аэропорты, библиотеки, супермаркеты, выставки, ярмарки) организуются автономные бюро находок, которые самостоятельно финансируются предприятием, на территории которого они расположены.

После попадания утерянной вещи в бюро находок, бюро находок регистрирует эту вещь и пытается по внешним признакам установить её владельца. Если это невозможно, утерянная вещь остаётся в хранилище бюро находок на определённое время, по истечение которого она либо уничтожается, либо переходит в муниципальную собственность. Вещи, перешедшие в муниципальную собственность, продаются на специальном аукционе, выручка от которого поступает в городскую казну. Найденные наличные деньги передаются на временное хранение в банк. Утерянные документы — в полицию. Утерянные иностранные паспорта направляются в консульство страны владельца паспорта.

При обращении за утерянной вещью в бюро находок, владелец должен доказать принадлежность ему найденной вещи. При выдаче утерянной вещи, бюро находок удерживает с владельца заранее установленную плату за регистрацию и хранение этой вещи [3].

В данной курсовой работе будет реализована информационная система «Бюро находок и потерянных вещей».

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Состояние вопроса

Потерять вещь легко, а вот вернуть ее всегда сложно. Так почему-то оно получается...

О первых «бюро» находок начали говорить еще тысячи лет назад в древней Греции и Риме. В Японии есть даже официальная дата открытия первого такого места, куда добропорядочные японцы приносили найденные ими чужие вещи – это 718 год. Современный же вид и функции бюро находок сформировались лишь в XIX веке.

Ежегодно люди во всем мире «сеют» десятки, а то и сотни миллионов вещей. Для примера, в 2002 году люди нашли и доставили в центр находок города Токио 23 миллиона долларов мелкой наличностью. 72% из этих средств вернули людям, которые смогли убедить полицию, что это их собственность.

Вообще, бюро находок подчиняются муниципальным органам власти. Также свои хранилища, как правило, есть в больших публичных местах – вокзалах, аэропортах, рынках, торговых центрах [6].

1. Poteryal

Сайт оборудован новейшими инструментами поиска любых предметов: драгоценностей, ключей, транспорта и даже животных. К тому же он охватывает широкую территорию поиска. Подав заявление, пострадавший в кротчайшее время получит информацию об утерянной вещи. Одним словом, теперь, если вы что-то нашли или потеряли, у вас есть возможность найти пропажу на единой платформе онлайн бюро находок.

Для поиска утерянных вещей не нужно регистрироваться. Достаточно выбрать подходящую вашему запросу категорию и ввести ключевою слово, например, ключи, паспорт, собака и нажать на кнопку «Поиск». Укажите место своего нахождения – страну, город, и на экране вы уже увидите результат своего поиска. Но если, например, вы не найдете в списке найденных своих вещей, - не отчаивайтесь. Подаете заявление о пропаже, которое будет быстро опубликовано для широкого просмотра. А чтобы подать заявление, уже нужно пройти регистрацию в социальной сети, заполнить форму заявления, указав приметы утерянной вещи или животного с предоставлением фотографии. Фотографии можно подавать разные, например, там, где вы вместе с животным, которое потерялось, или просто его снимок. Это значительно повысить шансы положительного поиска.

Для модераторов сайта очень важно, чтобы люди потерявшие и нашедшие вещи как можно быстрее связывались друг с другом. Это касается каждого! Если вы нашли утерянный кошелек, либо другие вещи, не проходите мимо, а быстро обращайтесь на сайт бюро находок poteryal.com. Следует отметить, что этот сервис очень простой и доступный в пользовании. Всего несколько кликов и перед вами высвечивается перечень потерянных вещей. Притом размещение об утерянный вещах, как и поиск новых, происходит всего за несколько секунд. Услугами данного сайта могут пользоваться граждане Украины, Белоруси, Казахстана. Для того, чтобы удобнее было пользоваться поданными объявлениями, можно зарегистрироваться. Это ускорит поиски.

Так что попав в стрессовую ситуацию, человеку не следует отчаиваться, а обратившись на данный сайт, можно быстро и оперативно устранить проблему [1].

Рисунок 1.1 – Онлайн бюро находок Poteryal

1.2 Актуальность и цель работы

Бренд советской эпохи – бюро находок – оказывается, не теряет своей актуальности. О потерянных вещах можно узнать в интернете, на аэровокзалах есть специальные комнаты забытых вещей, а некоторые пытаются заработать деньги на человеческой забывчивости.

Как ни странно, но бюро находок, так хорошо всем известные ещё с советских времен, актуальны до сих пор. Например, в крупных торговых центрах, вокзалах и аэропортах даже есть специальные комнаты, где хранятся забытые пассажирами вещи. К примеру, только в аэропорту «Кольцово» ежедневно находят не меньше пяти вещей, поэтому здесь только за полгода накопилось уже 240 находок. В основном, конечно, это книги, одежда, сотовые телефоны и другая бытовая техника. Однако один раз сотрудникам удалось обнаружить даже военную награду времён Великой Отечественной войны.

Утерянные вещи в таких комнатах хранят полгода. Если за это время они не находят хозяев, их списывают и утилизируют. Кроме того, сейчас в Екатеринбурге начали активно появляться и коммерческие бюро находок. Правда, существуют они пока только на деньги спонсоров и не приносят их учредителям никакого дохода. Но предприниматели не спешат отчаиваться: ведь в день горожане теряют довольно много различных вещей.

Конечно, если, к примеру, человек потерял крупную сумму денег, то вернуть её обратно вряд ли удастся. Но вот если это документы или даже животные – поискать их стоит в бюро находок. Шанс вновь обрести утраченное довольно велик.

Целью данной курсовой работы будет разработка информационной системы, которая позволит просто и легко искать и добавлять потерянные вещи в список.

Задачи, которые необходимо решить для реализации проекта:

1) изучить источники относительно потерянных вещей;

2) изучить подходы к реализации информационной системы бюро находок;

3) составить предполагаемый функционал приложения;

4) сформировать техническое задание(ТЗ), на основе которого будет разработано приложение;

5) выбрать и изучить технологии для реализации поставленных в ТЗ задач;

6) спроектировать дизайн приложения;

7) разработать приложение;

8) провести тестирование программного продукта;

9) написать руководство пользователя;

Предполагаемый функционал проекта:

1) создание характеристики;

2) возможность задать направление характеристики;

3) запись информации в отформатированный json-файл.

# 2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

2.1 Описание области применения и исходных данных приложения

Данное приложение предназначено для удобства работы при поиске и добавлении найденной вещи в систему для поддержания общей структурированности, так как персоналу больше можно будет не беспокоиться об многочисленных документациях и не отвлекаться от работы, чтобы предоставить клиенту список найденных вещей, потому данное приложение может применяться в любой сфере деятельности человека, где необходима работа с найденными либо потерянными вещами.

2.2 Требования к пользовательским интерфейсам

Основные требования к пользовательскому интерфейсу:

1. функциональность;
2. быстрое обучение пользователя;
3. обеспечение высокой скорости работы пользователя;
4. обеспечение защиты от человеческих ошибок;
5. понятность и логичность.

Пользовательский интерфейс должен быть ориентирован на самых обычных пользователей, быть достаточно понятным и удобным. Все элементы управления должны иметь свои всплывающие подсказки, чтобы упростить работу с приложением и его отдельными элементами.

На формах не должно быть лишних элементов.

Кнопки «Закрыть» и «Свернуть в трей» должны располагаться на привычных для пользователей местах в выпадающем меню. На дополнительных формах всё так же должно присутствовать.

Основные части функционального элемента «форма поиска в бюро ноходок (для администратора)» (формы для пользователя и администратора различаются лишь отсутствием возможности добавления новых вещей в программу), представленного на рисунке 2.1:

1) логин пользователя с выпадающим списком изменения базовой информации о пользователе и пароля, а также выхода из учётной записи;

2) текстовые поля для поиска;

3) выпадающий список по параметрам поиска;

4) кнопки: «Поиск» и «Добавление найденной вещи в список»;

5) кнопки «закрыть» и «свернуть в трей» на привычном для пользователя месте.

Рисунок 2.1 – Примерный макет формы поиска (для администратора)

Основные части функционального элемента «форма изменение данных о пользователе», представленного на рисунке 2.2:

1) поля для изменения никнейма и пароля (логин менять нельзя);

2) кнопка «Назад»;

3) кнопка «Сохранить изменения».

Рисунок 2.2 – Примерный макет формы изменение данных о пользователе

Основные части функционального элемента «форма добавление новой вещи», представленного на рисунке 2.3:

1. тестовые поля для ввода базовой информации о находке, а также информация о нашедшем и как с ним связаться;
2. кнопка «Назад»;
3. кнопка «Сохранить изменения».

Рисунок 2.3 – Примерный макет формы добавление новой вещи

Основные части функционального элемента «форма входа», представленного на рисунке 2.4:

1. текстовые поля для логина и пароля;
2. кнопка входа и регистрации.

Рисунок 2.4 – Примерный макет формы входа

Основные части функционального элемента «форма регистрации», представленного на рисунке 2.5:

1. текстовые поля для логина и пароля;
2. кнопки регистрации и отмены.

Рисунок 2.5 – Примерный макет формы регистрации

2.3 Требования к аппаратным, программным и коммуникационным интерфейсам

Для установки и работы приложения необходимо иметь вычислительную систему следующей минимальной аппаратной конфигурации:

1) процессор: 1.0 ГГц;

2) оперативная память: 512 Мб;

3) свободное место на жёстком диске: 500 Мб;

4) видеоадаптер: 64 Мб;

5) разрешение экрана 1920х1080 пикселей;

6) наличие основных устройств ввода-вывода (мышь, клавиатура, монитор).

Необходимое программное обеспечение:

1. операционная система Windows не ниже версии 7;
2. установленный .NET Framework не ниже версии 4.7.2.

2.4 Требования к пользователям продукта

Приложение предназначено для любой категории пользователей. Для пользования приложением необязательно обладать какими-либо специальными знаниями. Всё что необходимо пользователю, это умение пользоваться устройствами ввода-вывода, такими как компьютерная мышь и клавиатура.

2.5 Требования к адаптации на месте

Для установки приложения не требуются инсталляторы. Достаточно скачать директорию с программным обеспечением на свой компьютер и запустить приложение.

2.6 Функции продукта

1. Вход в систему: для успешного входа в систему «Бюро находок» необходимо ввести логин и пароль, пото нажать кнопку «Sign In»;
2. Регистрация в системе: для регистрации необходимо нажать кнопку «Sign Up», заполнить поля логина и пароля, потом подтвердить, после пользователь будет автоматически возвращен на форму входа.
3. Поиск потерянной вещи по параметру: после входа необходимо ввести необходимое значение в поисковой строке программы, далее в выпадающем списке выбрать колонку, где ожидается введённое значение и нажать кнопку «Search».
4. Изменение данных пользователя: для этого необходимо в правом верхнем углу нажать на текущих логин пользователя и в выпадающем меню выбрать пункт «PersonalInfo», после появится форма для изменения пароля и добавления никнейма (при создании аккаунта не задаётся, логин пользователя меняться не может), после изменения для сохранения нажать кнопку «Save».
5. Добавление найденной вещи (учётная запись администратора): необходимо нажать на кнопку «Add item» и заполнить все поля в появившейся форме, а потом нажать кнопку «Save» для сохранения найденной веди и дозаписи её в файл json.

2.7 Ограничения

1. Без регистрации использование функционала программы не возможна.
2. При некорректном введении логина либо пароля – невозможен вход в систему.
3. Посредством обычной регистрации можно завести лишь учётную запись пользователя.
4. Есть поля, что обязательны к заполнению, их содержимое не может быть опущено, при их не заполнении будет появляться уведомление, требующее заполнить все поля.
5. При не корректном заполнении полей появляется уведомление об их не корректности, а сами формы подсвечиваются красным цветом.

# 3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

# 3.1 Интегрированная среда разработки Visual Studio

Система Microsoft Visual Studio является самой сложной интегрированной средой разработки (integrated development enviroment - IDE), доступной для программистов в настоящее время. Она является результатом долгой истории развития языков программирования и интерфейсов и вобрала в себя достижения многих сред разработки программного обеспечения [7].

Разработана Visual Studio была в 1997 году и тогда это была первая среда разработки, в которой были собраны различные средства разработки ПО. Сама по себе, студия написана на языках С++ и С#.

Существует огромное множество сред разработки программного обеспечения, однако, почему именно Visual Studio является такой удобной при разработке приложения с использованием C подобного языка. Рассмотрим основные достоинства Microsoft Visual Studio перед другими средами разработки:

1. Огромная библиотека расширений которая постоянно увеличивается;
2. Настраиваемая панель и закрепляемые окна: возможность легко подстроить визуальный интерфейс сильно влияет на скорость и комфортабельность разработки;
3. Простой рабочий процесс и файловая иерархия: отличные помощники при проектировании приложения в стиле ООП;
4. Статистика мониторинга производительности в режиме реального времени: возможность на ходу проверить количество ресурсов которые тратит компьютер для поддержания работы программного обеспечения;
5. Инструменты автоматизации;
6. Легкий рефакторинг и вставка фрагментов кода: незаменимая помощь при разработке приложений;
7. Поддержка разделенного экрана;
8. Список ошибок, который упрощает отладку: возможность осуществлять быстрый и эффективный поиск ошибок, а так же упрощает их устранение.

По состоянию на 2019 год, Visual Studio и её аналог, созданный Microsoft занимают лидирующие позиции среди IDE. Это обуславливается доступностью и гибкостью этих сред разработки, широкими возможностями для расширения функционала под конкретные потребности разработчика за счет плагинов [3].

Рисунок 3.1 – Рейтинг сред разработки

Так же, Visual Studio лучшая среда разработки для работы на таких языках программирования как C# и С++.

# 3.2 Язык программирования С#

С# появился в 2000 году и был разработан компанией Microsoft, чтобы составить конкуренцию SunMicrosystem и их объектно-ориентированному языку Java. Он стал частью новой технологии Microsoft, названной .NET. В рамках этой технологии предусмотрена единая среда выполнения программ (Common Language Runtime, CLR), написанных на разных языках программирования. Одним из таких языков, основным в этой среде, и является C#. Этот язык был создан на основе языка С++ и стал одним из лучших объектно-ориентированных языков.

С# — простой, современный, объектно-ориентированный язык с безопасной системой типов, происходящий от С и С++. С# будет удобен и понятен для программистов, знающих С и С++. С# сочетает продуктивность Visual Basic и мощность С++.

Каковы же преимущества этого языка перед другими ООП языками, рассмотрим следующие достоинства:

1. Единицей компиляции является файл. Файл может содержать одно или несколько описаний типов: классов, интерфейсов, структур, перечислений, типов-делегатов, с указанием (или без указания) об их распределении по пространствам имен;
2. Структурированные типы: классы и интерфейсы, одномерные и многомерные массивы, строки, структуры, перечисления, несовместимые с целыми;
3. Типы подразделяются на ссылочные (классы, интерфейсы, массивы, делегаты) и типы-значения (элементарные типы, перечисления, структуры). Объекты ссылочных типов размещаются в динамической памяти (куче), а переменные ссылочных типов являются, по сути, указателями на эти объекты;
4. Автоматическая сборка мусора;
5. Обработка исключений;
6. Свойства - элементы классов (объектов), доступ к которым осуществляется так же, как и к полям, но реализуется неявно вызываемыми подпрограммами get и set;
7. События - элементы классов процедурного типа (делегаты), к которым вне класса, где они определены, применимы только операции += и –=, позволяющие добавить или удалить методы-обработчики событий для объектов данного класса [7].

На графике, представленном на рисунке 3.2 можно увидеть, что С# находится в тройке лидеров среди всех языков программирования. Это говорит о том, что язык является одним из самых признанных языков в мире на данный момент и огромное множество разрабатываемых приложений создается именно на этом языке программирования.

Рисунок 3.2 – Рейтинг языков программирования

Однако основной причиной выбора этого языка программирования является то, что на этом языке построена система построения клиентских приложений WPF. И именно в WPF реализуется используемый в курсовой работе архитектурный паттерн MVVM (Model-View-ViewModel).

# 3.3 Система для построения клиентских приложений Windows WPF

Windows Presentation Foundation (WPF) - это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. Она является частью платформы .NET Framework. WPF использует расширяемый язык разметки для приложений (XAML), чтобы предоставить декларативную модель для программирования приложений [6].

WPF развивается вместе с фреймворком .NET и имеет те же версии. Первая версия WPF 3.0 вышла вместе с .NET 3.0 и операционной системой Windows Vista в 2006 году. С тех пор платформа последовательно развивается. Последняя версия WPF 4.6 вышла параллельно с .NET 4.6 в июле 2015 года, ознаменовав девятилетие данной платформы.

Почему не Window Forms? На самом деле ответ довольно прост. WPF обладает колоссальным набором инструментов для работы с формами и графикой в отличие от WinForms. Также, WPF предоставляет некоторые функции для реализации ООП паттернов, что позволяет более эффективно разрабатывать приложения, используя именно эту систему, а не его предшественника – Windows Forms. И, как было сказано выше, именно в WPF реализуется архитектурный паттерн MVVM, используемый при разработке приложения. Преимущества WPF:

1. Для создания логики приложений используются стандартные языки .NET: C# и Visual Basic;
2. Обладает возможностью декларативного определения графического интерфейса с помощью специального языка разметки XAML, основанном на xml и предоставляет альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#/VB.NET;
3. Независимость от разрешения экрана: приложения разработанные в WPF легко масштабируются под экраны с разными разрешениями, так-как все элементы измеряются в независимых от устройства единицах;
4. Возможность создания трёхмерных моделей, привязка данных, стили, шаблоны, темы;
5. Богатые возможности по созданию различных приложений: поддержка мультимедиа, двухмерной и трёхмерной графики, создание анимации, возможность создания собственных элементов управления;
6. Аппаратное ускорение графики: WPF работает напрямую с DirectX, а потому повышается производительность и графика становится более плавной.

# 4 РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ «БЮРО НАХОДОК И ПОТЕРЯННЫХ ВЕЩЕЙ»

4.1 Входные и выходные данные приложения

Входные данные приложения: все данные о пользователе, список найденных вещей.

Выходные данные: вывод данных о пользователе на экран, отображение списка найденных вещей, запись данных о вещах и пользователях в txt-файлы.

4.2 Проектирование структуры приложения

Структурно приложение состоит из следующих классов:

1. User.cs – класс, описывающий пользователя.
2. FoundItem.cs – класс, описывающий найденную вещь.
3. WriteToFile.cs – класс, занимающийсяя записью информации о пользователях и найденных вещах в txt - файлы.
4. ExtractFromFile.cs - класс, занимающийсяя выборкой информации о пользователях и найденных вещах из txt – файлов.
5. DeleteData.cs - класс, занимающийсяя удалением информации о пользователях и найденных вещах из txt – файлов.
6. MainWindow.xaml.cs – класс, привязанный к форме входа.
7. SignUp.xaml.cs – класс, реализующий логику формы регистрации.
8. AdminPage.xaml.cs – класс, реализующий логику главной формы администратора.
9. UserPage.xaml.cs - класс, реализующий логику главной формы пользователя.
10. ChangeInfo.xaml.cs – класс, реализующий логику формы редактирования информации о пользователе.
11. Add\_Change\_Item.xaml.cs – класс, реализующий логику добавления вещи в систему.

Схема взаимодействия классов приведена на рисунке 4.1:

Рисунок 4.1 Диаграмма классов

* 1. Описание объектов и их взаимодействия для ООП

Запись информации о потярянной вещи и новом пользователе в файл

Первым делом создаём переменные для пути хранения файла (в данном случае файлы хранятся в корне проекта) и формируем строку для записи информации как о ользователе, так и о найденном предмете. Далее, проверяем уникальность данной записи (только при добавлении нового пользователя), после записывает строку в файл (создаём заново, если такого файла нет, в противном случае – дописываем информацию).

Ниже представлен фрагмент кода, реализующий запись информации в файлы. Вся запись реализована в отдельном классе WriteToFile.cs.

public void WriteUserData(User user)

{

string writePath = @"user.txt";

string text = user.Nickname+"#"+user.Login+"#"+user.Password+"#"+user.AdminRights;

if (CheckUser(text)) {

try

{

using (StreamWriter sw = new StreamWriter(writePath, true, System.Text.Encoding.Default))

{

sw.WriteLine(text);

}

Console.WriteLine("User Added!");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

public void WriteItemData(FoundItem item)

{

string writePath = @"items.txt";

string text = item.Name + "#" + item.Description +

"#" + item.Founder + "#" + item.Email + "#" +

item.Phone + "#" + item.Date.ToString().Replace("00:00:00","");

try

{

using (StreamWriter sw = new StreamWriter(writePath, true, System.Text.Encoding.Default))

{

sw.WriteLine(text);

}

Console.WriteLine("Item Added!");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

Полный код представлен в Приложении Б.1 - Файл «WriteToFile.cs».

Выборка информации о потярянной вещи и новом пользователе в файл

Первый шаг схож с записью, т.е. создаём переменную пути, где храниться файл. Открываем поток для считывания и в цикле занисим в лист данные из файла (при считывании списка потерянных вещей), либо сопоставляем пользовательские данные на входе с существиющими (для входа в приложение).

public void ReadUserData(User user)

{

string path = @"user.txt";

User tempUser = new User();

try

{

using (StreamReader sr = new StreamReader(path))

{

while (!sr.EndOfStream)

{

string tmp = sr.ReadLine();

string[] arr = tmp.Split('#');

tempUser.Nickname = arr[0];

tempUser.Login = arr[1];

tempUser.Password = arr[3];

tempUser.AdminRights = Convert.ToBoolean(arr[3]);

if(tempUser.Login==user.Login && tempUser.Password==user.Password)

GetUser = tempUser;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public void ReadItemsData()

{

string path = @"items.txt";

try

{

using (StreamReader sr = new StreamReader(path))

{

while (!sr.EndOfStream)

{

FoundItem tempItem = new FoundItem();

string tmp = sr.ReadLine();

string[] arr = tmp.Split('#');

tempItem.Name = arr[0];

tempItem.Description = arr[1];

tempItem.Founder = arr[3];

tempItem.Email = arr[3];

tempItem.Phone = arr[6];

tempItem.Date = Convert.ToDateTime(arr[6]);

items\_.Add(tempItem);

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

Полный код представлен в Приложении Б.2 - Файл «ExtractFromFile.cs».

# 5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

5.1 Аппаратные и программные средства создания и эксплуатации интернет-приложения

Для работы с приложением необходимо иметь вычислительную систему следующей базовой конфигурации:

1) процессор: 1.0 ГГц;

2) оперативная память: 512 Мб;

3) свободное место на жёстком диске: 200 Мб;

4) видеоадаптер: 512 Мб;

5) разрешение экрана 1920х1080 пикселей;

6) наличие основных устройств ввода-вывода (мышь, клавиатура, монитор).

Необходимое программное обеспечение:

1. операционная система Windows не ниже версии 7;
2. установленный .NET Framework не ниже версии 4.7.2.

Для работы с кодом программы вычислительную систему следующей базовой конфигурации:

1) процессор: 2.0 ГГц;

2) оперативная память: 2 Гб;

3) свободное место на жёстком диске: 2 Гб;

4) видеоадаптер: 512 Мб;

5) разрешение экрана 1920х1080 пикселей;

6) наличие основных устройств ввода-вывода (мышь, клавиатура, монитор).

Необходимое программное обеспечение:

1. операционная система Windows не ниже версии 7;
2. установленный .NET Framework не ниже версии 4.7.2;
3. Visual Studio не ниже версии 2020 года;
4. Установленная система для создания пользовательских приложений WPF;

5.2 Описание контрольных примеров

После запуска приложения отображается окно входа в учётную запись (рисунок А.1)

При отсутствии аккаунта необходимо зарегистрироваться, при нажатии кнопки «Sign Up» отобразится форма регистрации (рисунок А.3).

При успешно входе в учётную запись администратора либо пользователя отображается главное окно приложения (рисунок А.2 и рисунок А.4).

Пользователю доступны такие функции, как параметризированный поиск среди записей (рисунок А.8) и изменение информации о собственной учётной записи (рисунок А.6).

В учётной записи администратора также доступны такие функции, как добавление (рисунок А.7), изменение и удаление выделенной записи о найденном предмете (рисунок А.9).

Также стоит заметить, что при наведении указателя на любое текстовое поле тут же всплывают подсказки, что именно необходимо сюда ввести.

5.3 Руководство пользователя

1. Установка приложения.

Для работы приложения необходимо установить .NET Framework не ниже версии 4.7.2. Скачать его можно с официального сайта Microsoft и установить с помощью скачанного инсталлера.

Чтобы установить приложение на персональный компьютер, достаточно разархивировать архив с приложением в любую удобную директорию на компьютере.

2. Начало работы с приложением.

Для запуска приложения необходимо запустить исполняющий файл lostANDfound.exe.

Приложение откроется на странице входа с контролами: «Sign In», «Sign Up», «Login» и «Password».

3. Регистрация нового пользователя

Для регистрации нового пользователя в системе нужно выбрать соответствующий пункт в главном меню. Далее отроется окно для заполнения базовых данных вроде логина и пароля. При подтверждении произойдет проверка существующих пользователей, чтобы логин не повторялся, при успешной регистрации программа вернётся на форму входа.

4. Главное окно

При входе в систему программа проверяет уровень доступа пользователя, если это обычный пользователь, то окрывается стандартная форма пользователя с доступными функциями изменения общей информации о пользователе (изменять логин нельзя), а если это пользователя с уровнем доступа администратора, то к перечисленным выше функциям добавляются также добавление, изменение и удаление найденных предметов из таблицы.

# ЗАКЛЮЧЕНИЕ

В результате работы было разработана информационная система, которая, благодаря своему функционалу может стать незаменимым помощником для работников отдела бюро находок при добавлении, поиске либо удалении найденных вещей, а также позволяет пользователям легко и просто находить потерянные вещи в перечисленном списке.

Во время разработки приложения была изучена предметная область бюро находок. Была изучена документация языка программирования C#, фреймворка .NET Framework, системы для построения клиентских приложений WPF, изучена интегрированная среда разработки Visual Studio. Была изучена документация по объектно-ориентированному программированию. Все поставленные задачи были выполнены.

При разработке приложения, были подробно изучены методы работы с языком программирование С#, системой клиентских приложений WPF.

В будущем, планируется расширение функционала приложения и увеличение кросс-платформенности.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бюро находок перемещается в онлайн

URL: https://www.brd24.com/article/a-69678.html (дата обращения: 17.03.2020).

1. Бюро находок

URL: https://ru.wikipedia.org/wiki/Бюро\_находок#Организация (дата обращения: 17.03.2020).

1. Персональный блог… Самые популярные инструменты и средства разработки //

URL: https://igorosa.com/most-popular-tools-and-tools-development/ (дата обращения: 02.03.2020).

1. Потерянные вещи, или Как работает Бюро находок

URL:https://zik.ua/ru/news/2017/12/23/poteryannie\_veshchy\_yly\_kak\_rabotaet\_byuro\_nahodok\_1232305 (дата обращения: 17.03.2020).

1. Приступая к работе (WPF) // Microsoft – официальная домашняя cтраница

URL: https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/getting-started/ (дата обращения: 04.03.2020).

1. Работа с Visual Studio. // Professor Web

URL: https://professorweb.ru/my/programs/visual-studio/level1/ (дата обращения: 20.03.2020).

1. Язык программирования С# // Язык программирования С#-Критическая оценка

URL: http://www.uni-vologda.ac.ru/~c3c/articles/CScritics.htm (дата обращения: 17.03.2020).

ПРИЛОЖЕНИЕ А  
  
Экранные формы

Рисунок A.1 – Вход в приложение

Рисунок A.2 – Главная страница приложения (учётная запись администратора)

Рисунок A.3 – Регистрация пользователя

Рисунок A.4 – Главная страница приложения (учётная запись пользователя)

Рисунок A.5 – Выпадающий список для изменения информации о пользователе и выходе из учётной записи

Рисунок A.6 – Изменение данных о пользователе (никнейм и пароль)

Рисунок A.7 – Добавление новой найденной вещи

Рисунок A.8 – Параметризированный поиск

Рисунок A.9 – Выпадающее меню в таблицу с найденными вещами

# ПРИЛОЖЕНИЕ Б Фрагменты листинга

Листинг Б.1 – Файл «WriteToFile.cs»

public class WriteToFile

{

#region Private Fields

private Boolean userWritten\_;

private Boolean itemWritten\_;

#endregion

#region set-get Fields

public Boolean UserWritten

{

get

{

return userWritten\_;

}

set

{

userWritten\_ = value;

}

}

public Boolean ItemWritten

{

get

{

return itemWritten\_;

}

set

{

itemWritten\_ = value;

}

}

#endregion

#region Constructors

public WriteToFile() { }

#endregion

#region Private Methods

private Boolean CheckUser(String CheckString)

{

string path = @"user.txt";

string test = "";

try

{

using (StreamReader sr = new StreamReader(path))

{

test = sr.ReadToEnd();

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

if (test.Contains(CheckString))

return false;

else

return true;

}

#endregion

#region Public Methods

public void WriteUserData(User user)

{

string writePath = @"user.txt";

string text = user.Nickname+"#"+user.Login+"#"+user.Password+"#"+user.AdminRights;

if (CheckUser(text)) {

try

{

using (StreamWriter sw = new StreamWriter(writePath, true, System.Text.Encoding.Default))

{

sw.WriteLine(text);

}

Console.WriteLine("User Added!");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

public void WriteItemData(FoundItem item)

{

string writePath = @"items.txt";

string text = item.Name + "#" + item.Description +

"#" + item.Founder + "#" + item.Email + "#" +

item.Phone + "#" + item.Date.ToString().Replace("00:00:00","");

try

{

using (StreamWriter sw = new StreamWriter(writePath, true, System.Text.Encoding.Default))

{

sw.WriteLine(text);

}

Console.WriteLine("Item Added!");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

#endregion

}

Листинг Б.2 – Файл «ExtractFromFile.cs»

public class ExtractFromFile

{

#region Private Fields

private User user\_;

private List<FoundItem> items\_;

private Boolean userRead\_;

private Boolean itemRead\_;

#endregion

#region set-get Fields

public User GetUser

{

get

{

return user\_;

}

set

{

user\_ = value;

}

}

public List<FoundItem> foundItems

{

get

{

return items\_;

}

set

{

items\_ = value;

}

}

public Boolean UserRead

{

get

{

return userRead\_;

}

set

{

userRead\_ = value;

}

}

public Boolean ItemRead

{

get

{

return itemRead\_;

}

set

{

itemRead\_ = value;

}

}

#endregion

#region Constructors

public ExtractFromFile()

{

items\_ = new List<FoundItem>();

}

#endregion

#region Public Methods

public void ReadUserData(User user)

{

string path = @"user.txt";

User tempUser = new User();

try

{

using (StreamReader sr = new StreamReader(path))

{

while (!sr.EndOfStream)

{

string tmp = sr.ReadLine();

string[] arr = tmp.Split('#');

tempUser.Nickname = arr[0];

tempUser.Login = arr[1];

tempUser.Password = arr[3];

tempUser.AdminRights = Convert.ToBoolean(arr[3]);

if(tempUser.Login==user.Login && tempUser.Password==user.Password)

GetUser = tempUser;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public void ReadItemsData()

{

string path = @"items.txt";

try

{

using (StreamReader sr = new StreamReader(path))

{

while (!sr.EndOfStream)

{

FoundItem tempItem = new FoundItem();

string tmp = sr.ReadLine();

string[] arr = tmp.Split('#');

tempItem.Name = arr[0];

tempItem.Description = arr[1];

tempItem.Founder = arr[3];

tempItem.Email = arr[3];

tempItem.Phone = arr[6];

tempItem.Date = Convert.ToDateTime(arr[6]);

items\_.Add(tempItem);

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

#endregion

}