Содержание

1. Введение………………………………………………………..4
2. Постановка задачи……………………………………………..6
3. Требования к функциональным характеристикам…………..8
4. Требования к программным характеристикам……………....9
5. Проектирование системы………………………………….......10

5.1. Проектирование интерфейса ……………………………….10

5.2. Содержание приложения…………..…………………………11

5.3. Дизайн приложения……………………………………..……..12

5.4. Структура и удобство навигации приложения………..……..13

5.5. Грамматика и стиль в тексте…………………………………..13

5.6. Обновление и пополнение содержимого приложения………13

5.7. Качественное программирование………………….………….14

5.8. Интерактивность …………………………………………….…14

1. Проектирование базы данных …………………………...........15
2. Связывание базы данных mssql с приложением ...............…..16
3. Обзор страниц приложения………………………………….18
4. Тестирование приложения………………………………….….21
   1. Методы юзабилити-тестирования приложения………………21
   2. Тестовый макрос-сценарий…………………………………….25
   3. Тестирование инсталляции……………………………………..26
   4. Тестрование обновления ………………………………………27
   5. Тестирование деинсталляции ..............………………………..28
5. Выводы……………………………………………………………29
6. Список литературы……………………………………………….30
7. Лист замечаний ………………………………………………….31
8. Введение

Информационные технологии прочно вошли в нашу жизнь, доступность и объем информации, скорость ее обработки диктуют новые решения для множества сфер жизни – и частной, и общественной. Государство, наука, культура, образование, бизнес и все иные сферы человеческой жизнедеятельности сегодня немыслимы без организации свободного доступа к информации.

Основная идея информационной концепции в том, что вся информация должна быть организована в базы данных, а доступ к этим базам должен быть максимально упрощенным для пользователя. В рамках этой концепции преимущество оказывается у тех компаний и организаций, которые могут такой доступ предоставить.

В наше время большинство организаций имеют собственный вебсайт или десктопное приложение. А в очень распространенных случаях - и первое, и второе. Это является необходимым фактором, позволяющим расширить поле деятельности, привлечь новых клиентов, оперативно информировать их об изменениях, происходящих в организации, наладить быструю двустороннюю связь.

В данной работе представляется:

1. десктопное приложение для библиотек;

Цель данной работы - разработка приложения для библиотек, которая упрощает выполнения обязанностей для библиотекарей. Приложение основано на таком принципе: “Главная цель - счастливый пользователь приложения”.

Предмет исследования – теоретическое исследование понятия десктопного приложения, особенностей приложений коммерческой направленности, классификации средств создания данных программ, рассмотрение методов реализации создания десктоп-приложения.

Актуальность данной работы заключается в том, что стремительный прогресс в разработке все более новых технологий требует создания опережающих время программ для полноценной работы. По сути, библиотекарь должен иметь возможность вести учёт книг не в бумажном виде, а в электронной системе.

1. Постановка задачи

Разработка десктопного приложения для библиотек. Реализация возможности добавления, редактирования, удаления книг и читателей, поиск книг и читателей по разным критериям, выдачи книг и возврата их в библиотеку.

|  |  |
| --- | --- |
| Название структуры: | Министерство культуры Украины |
| Адрес, телефоны, контактные лица: | 095-111-11-11 |
| E-mail: | ukrminkult@gmail.com |
| Сфера деятельности структуры: | культура |
| География реализации товаров, услуг: | 24 города |
| Кто является основным потребителем товаров, услуг: | Организации, частные лица |
| Задача, которую должно решать приложение: | Улучшение возможностей работы в библиотеках |
| Тип программы: | библиотека |
| Основные разделы приложения: | Список всех книг, список читателей, список выданных книг |
| Частота предполагаемых обновлений приложения (раз в год, месяц, неделю, ежедневно): | Раз в неделю |
| Дальнейшее работа с приложением, (внесение обновлений) самостоятельно или поручается разработчикам сайта: | Самостоятельно |
| Языковые версии приложения: | Английский |
| Название приложения: | Library Helper |
| Стиль приложения (солидно, строго, просто и т.д.): | Просто, но стильно |
| Впечатление, которое должно произвести приложение на пользователя: | Свежесть, простор, надежность, опыт. |
| Какое основное сообщение необходимо довести до сознания потребителя: | Мы – надежная компания, которая оказывает качественные услуги, счастливый клиент - наш приоритет. |
| В какой примерно цветовой гамме должна происходить разработка приложения: | Светлая, фиолетовый, оранжевый, бежевый |
| На какую категорию пользователей должно быть ориентировано приложение: | Библиотеки |
| На какие географические регионы должно быть ориентировано приложение: | Все города (24) |

1. Требования к функциональным характеристикам

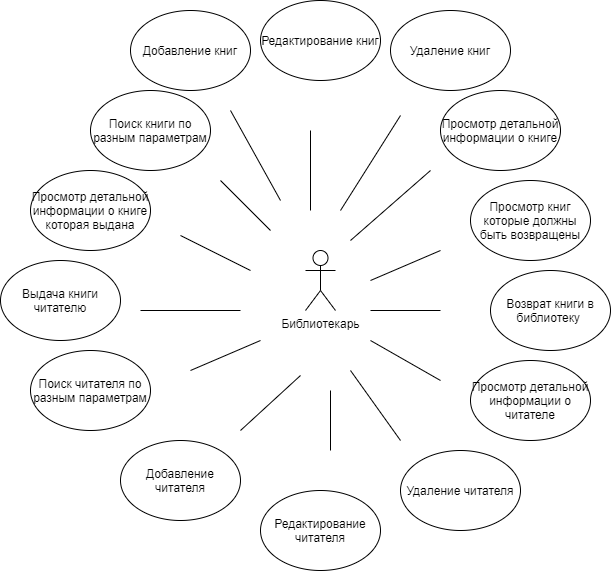


Рис. 3.1. Диаграмма пользователя

Рассмотрим функционал подробнее:

1. «Добавление, редактирование, удаление книг и/или читателя» - возможность управлять книгами и читателями
2. «Поиск книг и/или читателя по разным параметрам» - возможность поиска книг и читателей
3. «Просмотр детальной информации о книге и/или читателе» - возможность просмотреть детальную информацию
4. «Выдача книги читателю» - возможность выдать книгу читателю при условии если в списках выбраны соответствующие книга и читатель
5. «Возврат книги в библиотеку» - возможность вернуть книгу в библиотеку
6. «Просмотр книг, которые должны быть уже возвращены» - возможность просмотреть книги, которые должны быть уже возвращены
7. Требования к программным характеристикам

Требования к программным характеристикам клиента:

* OS – Windows XP и выше;

Требования к программным характеристикам сервера:

* Поддержка MsSQL
* OS: Linux/Windows

Структура приложения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Список всех книг в библиотеке | |  | Список читателей | |
|  | Список книг, которые выданы читателям | | |  |

1. Проектирование системы

5.1. Проектирование интерфейса

Данная программа является средством управления для определенного круга пользователей (библиотекарей), что требует определенных принципов и подходов к разработке интерфейса.

Разработчик стремится создать удобное, простое и понятное окружение, в котором клиент чувствовал бы себя комфортно.

По вопросу цветовой гаммы решением стали неяркие, и не резкие цвета. Задний фон приложения не только не отвлекает от основного содержимого, но и сочетается с неактивными пунктами главного меню.

Управление с помощью мыши:

1). Все действия происходят по одинарному клику левой кнопки мыши.

Есть несколько общепринятых правил создания хорошего приложения:

5.2. Содержание приложения

Приложение должно содержать информацию (контент). Прочтите ещё раз и вдумайтесь - оно должно содержать ИНФОРМАЦИЮ. Именно из-за неё пользователи стремятся в Интернет и не только. Чем больше конкретной информации, тем легче собрать большую аудиторию его посетителей.

5.3. Дизайн приложения

Хотя, есть поговорка, что по одёжке встречают, я сделал этот пункт вторым после информации. Но это совсем не значит, что дизайн программы менее важен. Первое, с чего начинают оценку, это внешний вид, а уже потом делают выводы, изучая содержимое. качество (уровень) оформления приложения прямо показывает уровень уважения к клиенту, существующий в компании. История имиджевой рекламы уже существует около 200 лет (не у нас, конечно), а это многое значит. Десктопное приложение - лицо структуры, и по качеству оформления будут судить о вас и будут сравнивать вас с вашими конкурентами. Кроме аккуратного исполнения, существует ещё множество правил, которые необходимо учитывать. Есть технические тонкости, например - вес (размер) графических файлов, оправданность применения некоторых технологий. А есть более скрытые: эстетическая ценность, креатив (самое непонятное слово и объект спекуляции на понимании терминов), правильность и совместимость цветового баланса, грамотность пропорций и перспектив по объёму и освещению. Плюс, немного психологии - правильное ассоциативное восприятие графического оформления, в соответствии с тематикой приложения и и его задачами. Часто, этот вопрос имеет большее значение, чем кажется на первый взгляд.

5.4. Структура и удобство навигации приложения

Если ценность информации можно сравнить с бриллиантом, дизайн - с его оправой (хотя, иногда, бриллианты являются только частью украшения), то структуру и навигацию можно сравнить с огранкой камня (правда, при огранке, камень теряет в весе). Хорошая программа - всегда тщательно продумана. Пользователь должен легко понимать структуру и специфику предложения компании по навигации, а инструменты навигации должны быть понятны. Хорошо, когда навигация учитывает не только разделы приложения, но и удобство работы с каждой его страницей. Хорошее приложение не должно иметь "пустых" страниц с сообщениями о ведущихся работах по наполнению. Вы можете прочитать статью, которая более подробно затрагивает эту тему.

5.5. Грамматика и стиль в тексте

Ошибки, опечатки и плохая литературная стилистика в тексте - недопустимы в приложении. Не соблюдение правил вёрстки текста, подбора шрифтов и их размера - также снижают его оценку пользователем.

5.6. Обновление и пополнение содержимого приложения

Чтобы пользователи постоянно возвращались к вам, приложение должно быть "живым". Обновление устаревшей информации и пополнение новой должно проводиться регулярно. Очень хорошо, когда это поручено конкретному человеку или коллективу. Часто, это приводит к регулярным расходам - оплата квалифицированного специалиста или регулярные платежи за "поддержку" приложения(у нас эта проблема решена средствами разработанными нами программами). Это полностью законченные решения, которые позволяют нашим заказчикам получить максимальную независимость.

5.7. Качественное программирование

Мир разнообразен. И Интернет тоже. Существуют различные броузеры и множество их версий, у пользователей разные размеры мониторов и разное разрешение экранов, есть различные кодировки текста и разные шрифты. Но всё это не должно быть помехой между пользователем и вашим сайтом или приложением. Хорошее приложение не должен содержать лишний код, а программные решения должны работать без ошибок и "помогать" обычному пользователь не ошибаться.

5.8. Интерактивность

Интернет позволяет вести общение в обоих направлениях. Пользователь всегда может обратиться с просьбой или вопросом, если есть адреса электронной почты. А своевременный ответ оставит хорошее впечатление о сайте или приложении и его владельцах. Различные программные решения в вашем приложении, могут ещё больше способствовать повышению интерактивности.

1. Проектирование базы данных

При проектировании системы было разработано и создано множество таблиц для удобного хранения и управления данными.

Был применен MsSQL. Использован C# и Entity Framework Core.

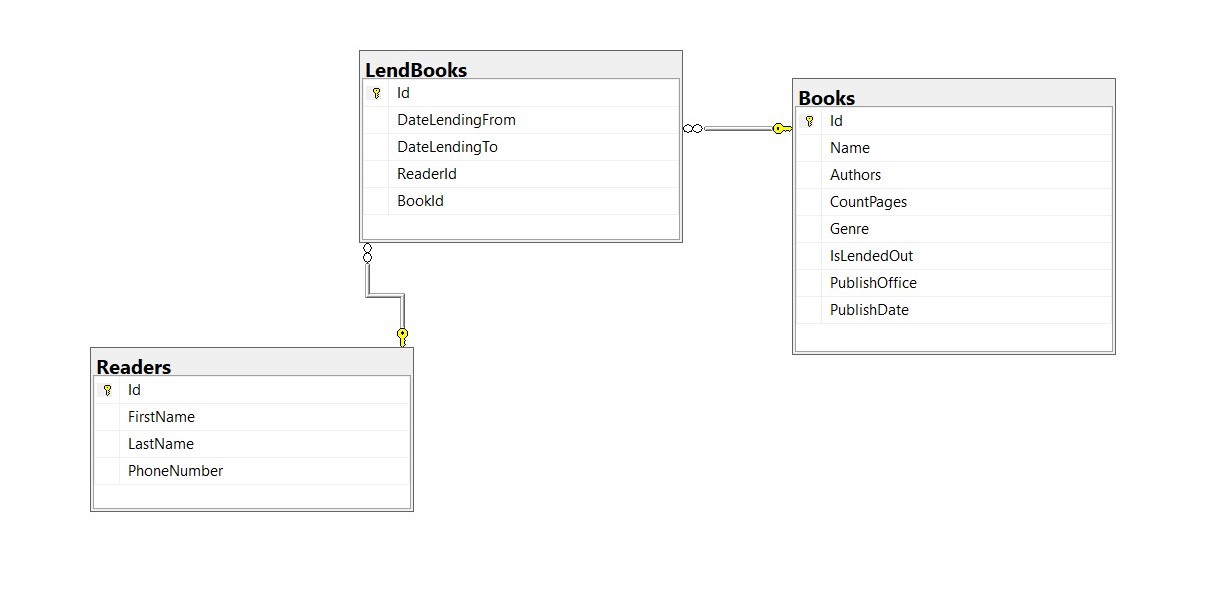


рис. 6.1 Диаграмма таблиц базы данных Library Helper

Подробнее о таблицах бд. Library Helper

1. таблица Books– информация о всех книгах в библиотеке;
2. таблица LendBooks– информация о читателях;
3. таблица Readers– информация о книгах, которые выданы читателям;
4. Связывание базы данных msql с приложением

Для работы з базой данных я использовал пакет Entity Framework Core.

1) Подключение к БД

Прежде чем начать работу с базой данных необходимо установить соединение с сервером, на котором она установлена.

Для этого я реализовал класс StoreContext, который наследует класс DbContext. С помощью метода OnConfiguring(optionsBuilder) я сконфигурировал подключение к базе, задав строку подключения к ней.

optionsBuilder - объект класса DbContextOptionsBuilder.

2) Выполнение SQL – запроса

После выполнения подключения к базе данных можно отправлять SQL – запросы для их дальнейшего выполнения на сервере базы данных. Стоит отметить, что для выполнения разных запросов я использовал Linq. Также все действия с базой происходят асинхронно.

Запросы имеют свою специфику реализации. Я опишу два - добавление объекта в базу и получение всех объектов с условием из базы.

Добавление реализовано с помощью такой функции:

public virtual async Task Create(T entity)

{

db.Entry(entity).State = EntityState.Added;

await db.SaveChangesAsync();

},

db.Entry(entity).State, где entity - это сущность для добавления, EntityState.Added - состояние добавляемого объекта, который я получаю с помощью метода db.Set<>(), где db - объект моего класса StoreContext.

Получение с условием имеет другую специфику:

public virtual async Task<IEnumerable<T>> GetAll(Func<T, bool> predicate)

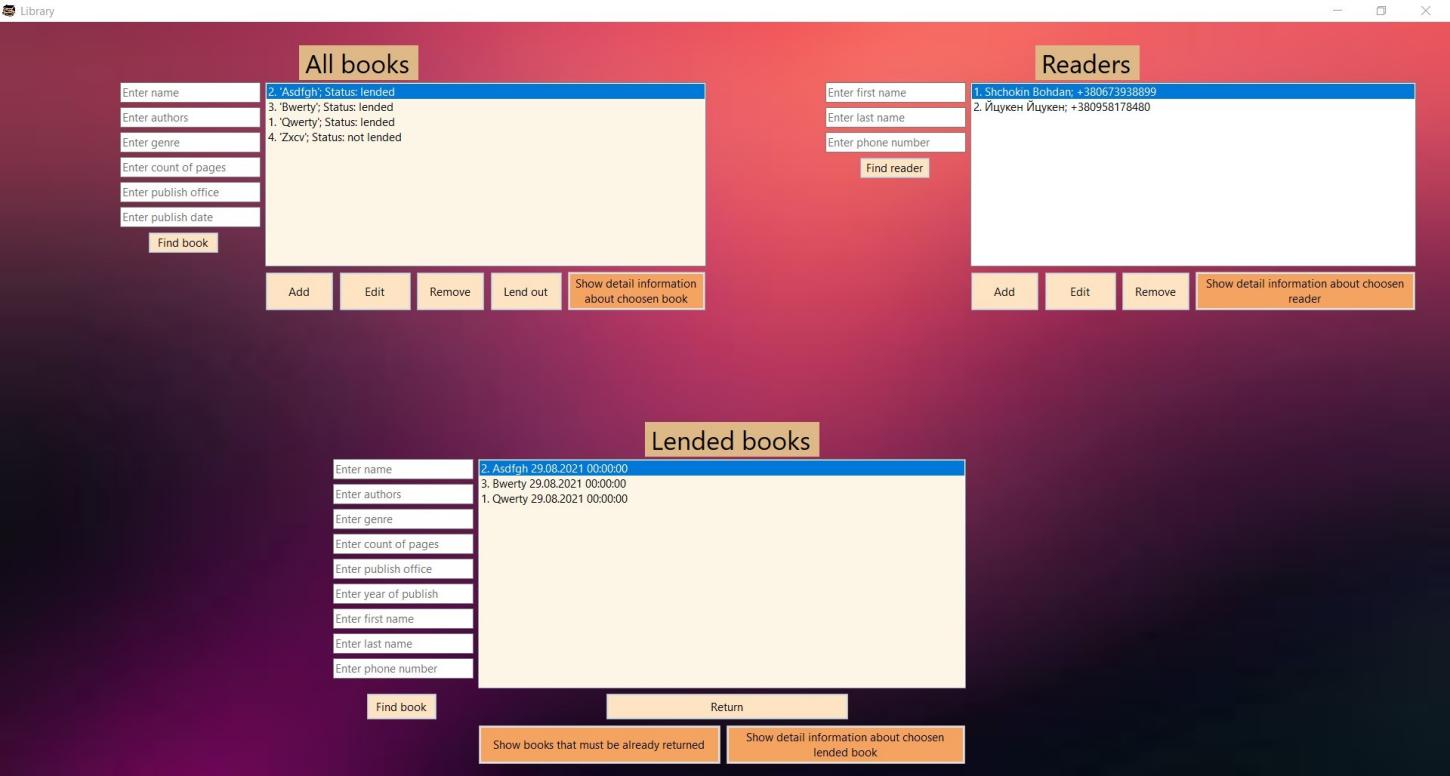
{

return await Task.FromResult(Table.Where(predicate).ToList());

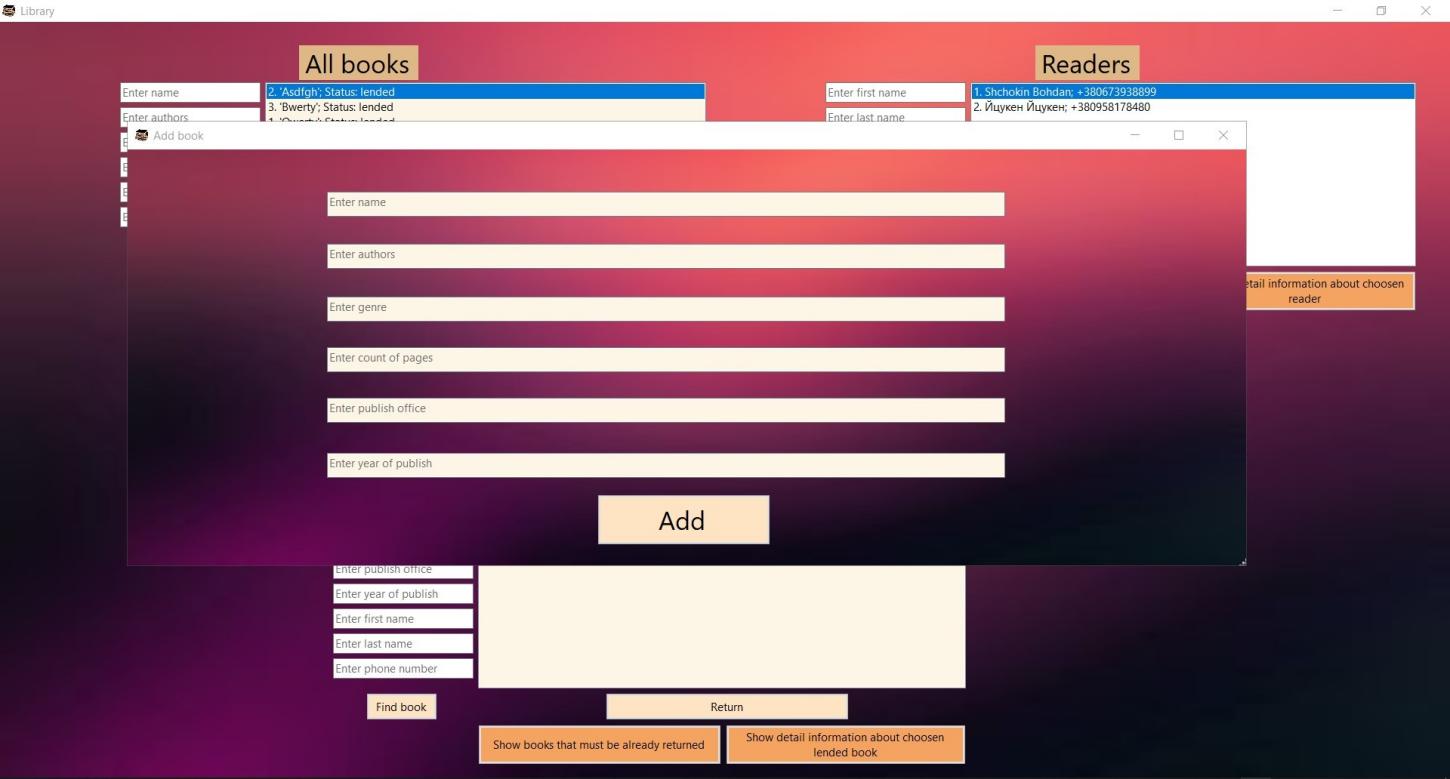
},

Table - объект типа DbSet<>, который я получаю с помощью метода db.Set<>(), predicate - делегат, который принимает условие и возвращает результат true or false. Виртуальным я сделал этот метод для того, чтобы в дальнейшем можно было его перегружать.

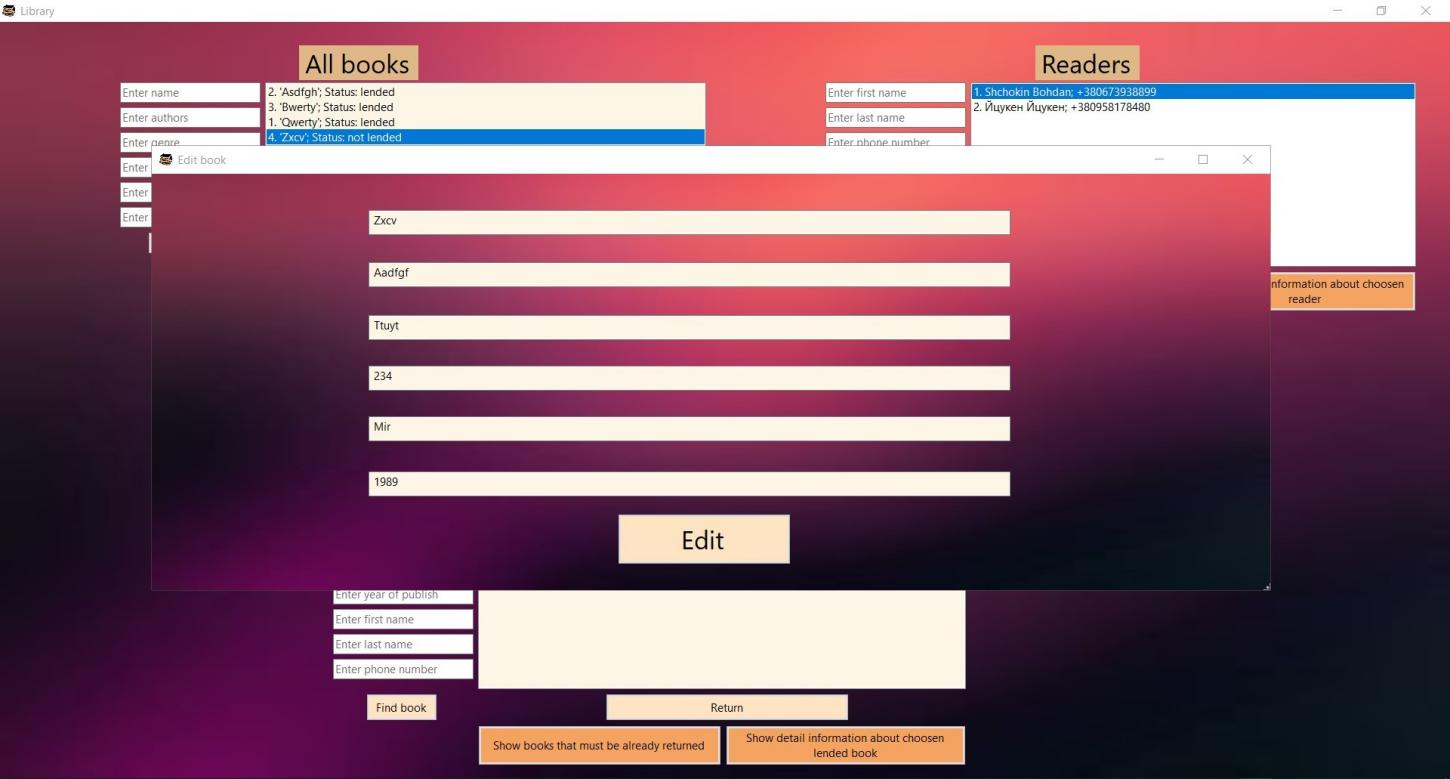
1. Обзор окон приложения



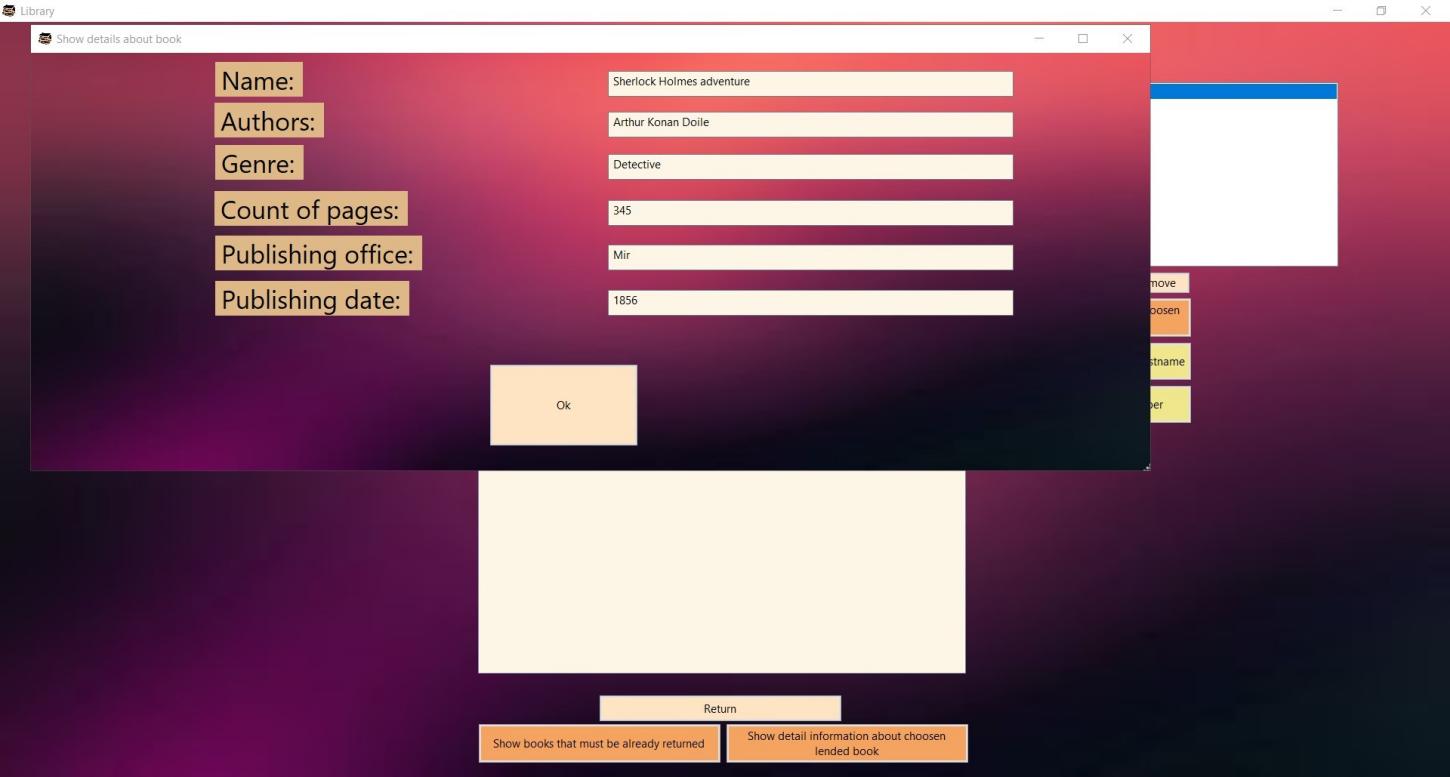
Главное меню



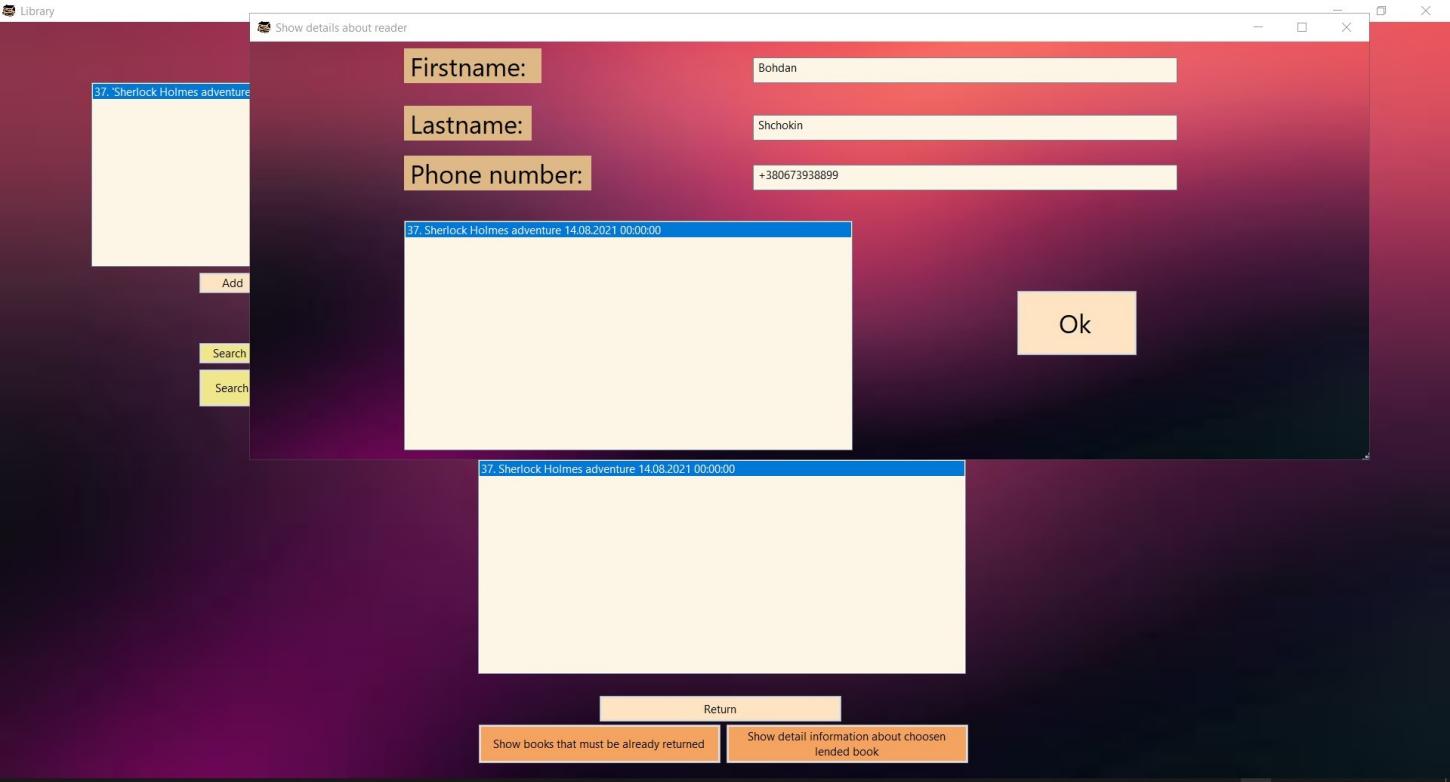
Окно добавления



Окно редактирования



Окно просмотра детальной информации о книге



Окно просмотра детальной информации о читателе

1. Тестирование и отладка программы

9.1 Методы юзабилити-тестирования приложения

Юзабилити-тестирование — это эксперимент, выполняемый с целью определения, насколько хорошо люди могут использовать некоторый искусственный объект (такой как приложение, пользовательский интерфейс или устройство) для его предполагаемого применения, то есть юзабилити-тестирование измеряет юзабилити объекта. Юзабилити-тестирование сосредоточено на определённом объекте или небольшом наборе объектов, в то время как исследования взаимодействия человек-компьютер в целом — формулируют универсальные принципы. Юзабилити-тестирование — метод оценки удобства продукта в использовании, основанный на привлечении пользователей в качестве тестировщиков, испытателей и суммировании полученных от них выводов.

Тестирование проводится с участием нескольких человек из целевой аудитории, так называемых респондентов. Для проведения тестирования достаточно 4-6 человек. Существует правило 80/20, которое гласит, что 20% пользователей дают 80% результата. Поэтому такое количество респондентов максимально эффективно с точки зрения экономии времени и затрат. Существует несколько методов оценки ранних концепций.

Метод карточной сортировки — это классификационный метод, при котором пользователи сортируют различные элементы разрабатываемого приложения по нескольким категориям.

Для проведения карточной сортировки создаётся список параметров, которые предполагается подвергнуть классификации, после чего каждый из указанных параметров выписывается на отдельной карточке. Карточки предъявляются пользователям, которых инструктируют сгруппировать наиболее логичным, по их мнению, образом, и обязательно дать названия этим группам.

Прямая карточная сортировка позволяет выявить то, как пользователь категорирует понятия. Это понятия лягут в основу названий навигации.

Метод обратной карточной сортировки — это верификация прямой карточной сортировки, то есть проверка того, что спроектированная структура навигации понятна пользователю.

Обратную карточную сортировку можно проводить как списком (текстовый способ), так и карточками (графический способ). В случае с текстовым способом пользователю даётся верхний список пунктов меню, так называемый топ левел, и даётся задание: например, найти в библиотеке книгу. И респондент должен сказать где, по его мнению, расположена данная вещь, после чего раскрывается меню второго уровня, то есть даётся второй список.

Графический способ предполагает несколько иное: пользователю даётся список меню и группа карточек, которые он должен разложить по пунктам меню таким образом, который по его мнению верный.

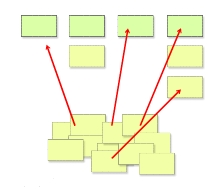


Рис.9.1 Метод обратной карточной сортировки

Метод тестирования ожиданий — тест на понимание пользовательских процессов, происходящих в системе. Каждый пользователь представляет, что получится после совершения какого-либо его действия, ожидания формируются на основе предыдущего опыта взаимодействия с системой. Проводится как самостоятельное тестирование, так и включённое в процесс юзабилити-тестирования. Процедура тестирования следующая: пользователю выдаётся обязательно распечатанный вариант интерфейса, после чего задаются вопросы об ожидаемом поведении того или иного элемента: «Что вы ожидаете увидеть, когда нажмёте на него?», «Что произойдёт, если нажать на эту кнопку?»

В юзабилити-тестировании есть понятие «парадокс активного пользователя». Это означает, что пользователь производит действие раньше окончания вопроса, поэтому при тестировании ожиданий пользователю необходимо предоставлять распечатанный вариант интерфейса.

При использовании метода тестирования ожиданий выявляются дефекты. Дефекты — это расхождение ожиданий и запланированного поведения системы.

Эти методы позволяют провести тестирование на самом раннем этапе, ещё не имея графического дизайна. В результате этих тестирований исправляются критические ошибки. Критическая ошибка — это ошибка респондента из-за непонимания структуры интерфейса, которая привела к другим ошибкам.

Метод оценки восприятия дизайна позволяет понять вызывает ли дизайн целевые эмоции. Дизайнеру вместе с макетом интерфейса выдаётся список прилагательных, которые должны характеризовать дизайн сайта, то есть какие эмоции дизайн должен вызывать. Как правило, этот список эмоций можно найти в брендбуке любой компании, это задача маркетологов. После чего, готовый интерфейс в нескольких вариантах цветового оформления предоставляется респондентам, и также даётся шкала с этими прилагательными, характеризующими общее впечатление, которое они должны оценить. Например, молодой - старый, красивый - не красивый, интересный - не интересный, удобный - неудобный и т. д.

В итоге, выбирается тот дизайн, который вызывает у пользователей нужные эмоции. Если по сумме ожиданий они оказываются примерно равными, выбирается по лучшим ожиданиям: ключевые плюс дополнительные.

Также существуют и более дорогостоящие методы оценки и тестирования готового интерфейса: с использованием специального программного обеспечения и оборудования. Например, Eye tracking.

Метод Eye tracking

Eye tracking – это система, используемая юзабилити-специалистами для регистрации движения глаз. Этот метод осуществляется с использованием веб-камеры и специального программного обеспечения. Таким путём составляется тепловая карта интерфейса. Тепловая карта — карта приложения, на которой отмечены наиболее горячие участки - это места, где дольше всего фиксируется взгляд пользователей. Этот метод хоть и является эффективным, в плане понимания какие зоны попадают во внимание пользователя, а какие наоборот — проблемные, однако, он не даёт объяснения, почему так происходит.

Такое тестирование одно из самых дорогих, потому что наиболее ценную информацию можно получить только от реальных пользователей, наблюдая за их работой с вашим приложением — а такие исследования требуют дорогостоящей инфраструктуры и времени, их сложно автоматизировать. О тестировании и самом "usability" уже писалось на страницах hostinfo.ru — «Юзабилити – новое полезное слово в разработке» и статья именно об исследовании и оценке: «Оцениваем юзабилити».

9.2. Тестовый макрос-сценарий

Это запись специального тестового макроса-сценария, который имитирует действия реальных пользователей. Потом программным образом создаются несколько виртуальных пользователей (тысячи и более при серьезном тестировании) и они «ходят» в приложение, проверяя его работу. За этим всем следит менеджер (обычный, человек?), строя на основе полученных данных отчет о работе исследуемого приложения.

Такие тест-комплексы могут быть довольно большими — даже в прямом смысле, объем дистрибутивов составляет сотни мегабайт. Как пример — комплексное решение для тестирования приложений Empirix — e-TEST suite. e-TEST может использоваться для комплексного тестирования приложений, созданных с использованием Microsoft .NET, а также обычные приложения. Комплекс включает в себя решения для разностороннего тестирования — от интерфейса до тестирований нагрузочной способности, безопасности и даже проведение тестов удалённых частей приложения в реальных условиях. Конкретно, для тестирования интерфейса, в состав e-TEST suite входит компонент под названием e-Tester — в оригинале он называется «Automated Regression and Functional Testing Tool for Web Applications (Record/Playback)». Это компонент как раз и предназначен для записи определённой последовательности действия и воспроизведение их впоследствии, имитируя пользователей. Другой компонент, e-Monitor, в реальном времени собирает статистику тестирования и информацию о возникших ошибках, а также позволяет планировать тесты (например, если тестируется удалённая часть приложения, то его можно тестировать в разные периоды времени).

9.3 Тестирование инсталляции

Выполняя тестирование установки проверяется:

1. Запускается ли программа после установки
2. Расположение программы в файловой системе по-умолчанию
3. Расположение программы в файловой системе если путь сохранения изменен пользователем
4. Наличие ярлыков на рабочем столе
5. Есть ли установленный компонент в меню Пуск  > Программы
6. При установке обратить внимание на издателя
7. Установка программы для текущего пользователя/для всех пользователей компьютера
8. Установка пользователем с правами админа
9. Установка пользователем без прав админа

9.4. Тестирование обновлений

Для тестирования обновлений специально устанавливают старую версию программы, она сразу же находит обновления и обновляется. Выполняя тестирование обновлений нужно:

1. Проверить что после установки обновлений данные пользователя не были повреждены
2. Проверить что все созданные ранее пользователем файлы остались доступными

9.5. Тестирование деинсталляции

Выполняя тестирование удаления проверяем:

1. Файлы должны удалиться
2. Ярлык с рабочего стола исчез
3. Удалена ли запись из меню Пуск > Все программы
4. Выполняем команду %userprofile% через командную строку, чтобы открыть личную папку текущего пользователя. Убеждаемся, что нет папок с названием программы
5. Выводы

Из полученных результатов видно, что наиболее приемлемым способом для создания подобного рода десктопных приложений по праву является язык программирования C#, с помощью всевозможных технологий, таких как Windows Forms, ADO.NET и использовании разных сторонних пакетов. Приложение обладает интерактивным контентом, что позволяет пользователю в полной мере ознакомиться с актуальными услугами, выбрать интересующие себя условия работы и принять решение.

На сегодняшний день в своей области десктопное приложение бизнес плана немного проигрывает в актуальности бизнес-сайту. Бизнес-сайт отличается тем, что может работать на слабых ПК. В то время как десктоп версия предназначена для более производительных ПК.

Однако, основная задача как бизнес-сайта, так и десктопного приложения - упростить выполнения обязанностей библиотекарей.

12.Список литературы

* 1. Wikipedia
  2. Руководство по C#
  3. <https://metanit.com> - сайт о программировании
  4. https://qaevolution.ru/625-2/ -методы тестирования
  5. Свободные интернет источники

13.Лист замечаний

|  |  |
| --- | --- |
| Ф.И.О. | Суть замечаний, оценка и подпись |
|  |  |