Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

**КУРСОВАЯ РАБОТА**

**ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «УЧЁТ ВЫДАЧИ И ВОЗВРАТА КНИГ»**

**ПМ.05 «ПРОЕКТИРОВАНИЕ И разработкА ИНФОРМАЦИОННЫХ СИСТЕМ»**

**МДК 05.03 «ТЕСТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ»**

**09.02.07 Информационные системы и программирование**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  | **/** | А.А. Храмков | |
|  | *подпись* |  | *И.О. Фамилия* | |
| \_\_.\_\_.2022 г. |  |  |  | |
| Оценка выполнения и защиты курсовой работы | | | | |  | |
|  | | | | |
| Руководитель |  | **/** | Н.К. Коровина | |
|  | *подпись* |  | *И.О. Фамилия* | |
| \_\_.\_\_.2022 г. |  |  |  | |

Тольятти, 2022

Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

|  |  |  |
| --- | --- | --- |
|  |  | Утверждаю:  Заместитель директора по УР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.С. Киронова  *« » 2022 г.* |

**ЗАДАНИЕ**

на курсовую работу

**по ПМ.05 Проектирование и разработка информационных систем модуля, выполняемой в рамках МДК.05.03 Тестирование информационных систем**

студента группы ИСП-31

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* Храмкова Александра Александровича*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия Имя отчество студента*

|  |  |
| --- | --- |
| Тема курсовой работы: **«**Учёт выдачи и возврата книг» |  |

1. Содержание задания:

1.1 Тестирование информационной системы.

1. Исходные данные:

Исходные данные для практической реализации автоматизированной информационной системы (АИС) берутся из различных информационных источников (Интернет-ресурсы, печатные издания, периодика и др.).

1. Содержание курсовой работы

Введение

1 Анализ методов тестирования

1.1 Критерии и принципы тестирования

1.2 Методы тестирования

2 Тестирование информационной системы (название)

2.1. Разработка тестовой документации (тест-дизайн)

2.2. Разработка тестовых сценариев

Заключение

Список использованных источников

Дата выдачи задания: « » 2022 г.

Дата сдачи работы на отделение: « » 2022 г.

Руководитель курсового(ой) проекта(работы) \_\_\_\_\_\_\_\_\_\_\_\_\_ / Н.К. Коровина\_\_\_\_\_\_

подпись расшифровка подписи

**КАЛЕНДАРНЫЙ ПЛАН**

выполнения курсовой работы

Студентом 3 курса группы ИСП-31

По теме Тестирование информационной системы «Учёт выдачи и возврата книг»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  этапа  работы | Содержание этапов работы | Плановый срок выполнения этапа | Планируемый объем выполнения  этапа, % | Отметка  о  выполнении  этапа |
| 1 | Выбор, обоснование темы и объекта исследования | Январь 2022 | 5% |  |
| 2 | Утверждение темы, согласование плана. Введение, библиография | Январь 2022 | 10% |  |
| 3 | Изучение и анализ информационных материалов по теме | Февраль 2022 | 15% |  |
| 4 | Обоснование актуальности выбранной темы применительно к профессиональной деятельности (введение) | Февраль 2022 | 20% |  |
| 5 | Изложение материала основной части по теме курсовой работы | Февраль 2022 | 20% |  |
| 6 | Подведение итогов проведенного анализа, формулировка выводов УИР применительно к профессиональной деятельности (заключение) | Март 2022 | 20% |  |
| 7 | Оформление работы и сдача на проверку | Март 2022 | 10% |  |
| 8 | Защита работы |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  | **/** | А.А. Храмков |
|  | *подпись* |  | *И.О. Фамилия* |
| 17.01.2022 г. |  |  |  |
| Руководитель |  | **/** | Н.К. Коровина |
|  | *подпись* |  | *И.О. Фамилия* |
| 17.01.2022 г. |  |  |  |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc101374556)

[1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ 8](#_Toc101374557)

[1.1 Критерии и принципы тестирования 8](#_Toc101374558)

[1.2 Методы тестирования 10](#_Toc101374559)

[2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ РАСЧЕТ СТОИМОСТИ УСЛУГ, ПРЕДОСТАВЛЯЕМЫХ КЛИЕНТАМ 15](#_Toc101374560)

[2.1. Разработка тестовой документации (тест-дизайн) 15](#_Toc101374561)

[2.2. Разработка тестовых сценариев 17](#_Toc101374562)

[ЗАКЛЮЧЕНИЕ 19](#_Toc101374563)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc101374564)

# ВВЕДЕНИЕ

Первые программные системы разрабатывались в рамках программ научных исследований или программ для нужд министерств обороны. Тестирование таких продуктов проводилось строго формализованно с записью всех тестовых процедур, тестовых данных, полученных результатов. Тестирование выделялось в отдельный процесс, который начинался после завершения кодирования, но при этом, как правило, выполнялось тем же персоналом.

В 1960-х много внимания уделялось «исчерпывающему» тестированию, которое должно проводиться с использованием всех путей в коде или всех возможных входных данных. Было отмечено, что в этих условиях полное тестирование программного обеспечения невозможно, потому что, во-первых, количество возможных входных данных очень велико, во-вторых, существует множество путей, в-третьих, сложно найти проблемы в архитектуре и спецификациях. По этим причинам «исчерпывающее» тестирование было отклонено и признано теоретически невозможным.

В начале 1970-х годов тестирование программного обеспечения обозначалось как «процесс, направленный на демонстрацию корректности продукта» или как «деятельность по подтверждению правильности работы программного обеспечения». В зарождавшейся программной инженерии верификация ПО значилась как «доказательство правильности». Хотя концепция была теоретически перспективной, на практике она требовала много времени и была недостаточно всеобъемлющей. Было решено, что доказательство правильности — неэффективный метод тестирования программного обеспечения. Однако, в некоторых случаях демонстрация правильной работы используется и в наши дни, например, приёмо-сдаточные испытания. Во второй половине 1970-х тестирование представлялось как выполнение программы с намерением найти ошибки, а не доказать, что она работает. Успешный тест — это тест, который обнаруживает ранее неизвестные проблемы. Данный подход прямо противоположен предыдущему. Указанные два определения представляют собой «парадокс тестирования», в основе которого лежат два противоположных утверждения: с одной стороны, тестирование позволяет убедиться, что продукт работает хорошо, а с другой — выявляет ошибки в программах, показывая, что продукт не работает. Вторая цель тестирования является более продуктивной с точки зрения улучшения качества, так как не позволяет игнорировать недостатки программного обеспечения.

При изучении предметной области, выяснилось, что информационная система «Учёт выдачи и возврата книг» казалось, проста в реализации, но это мнение ошибочно. Для реализации данного программного продукта следует обратиться к взаимодействию с базами данный MSSQL.

Объектом исследования курсовой работы является процесс учёта выдачи и возврата книг в библиотеке, где предметом исследования процесса является ИС «Учёт выдачи и возврата книг».

Вышеизложенное в целом на теоретико-методологическом уровне определило проблему настоящего исследования: тестирование программного модуля «Учёт книг» в программе MS Visual Studio 2022.

Целью курсовой работы является изучить процесс тестирования как составляющую процесса обеспечения качества разработки ПО, а также теоретически обосновать основные положения данного процесса и проверить их практически на основе разработанной информационной системы «Учёт выдачи и возврата книг».

Для достижения поставленной цели были определены следующие задачи:

# проанализировать методы тестирования;

# изучить критерии и принципы тестирования;

# изучить методы тестирования;

# протестировать информационную систему;

# разработать тестовую документацию;

# разработать тестовые сценарии.

Для тестирования информационной системы будут применяться методы: тестирование деятельности предприятия с выявлением его функции, а также функциональное тестирование системы.

Практическая значимость заключается в использовании протестированной системы на различных предприятиях по «Учёт выдачи и возврата книг».

Структура работы соответствует логике исследования и включает в себя введение, две главы, заключение, список использованной литературы.

# 1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ

## 1.1 Критерии и принципы тестирования

Из множества принципов, которые наверно для себя выделил каждый тестировщик занимающийся тестированием ПО есть основные по мнению сообщества тестировщиков, которые и хотелось бы выделить:

1) Исчерпывающее тестирование невозможно ни одним из тестировщиков. Думаю что все понимают что протестировать все возможные случаи и комбинации просто невозможно, конечно если это не тривиальный случай.

Все случаи просто не могут быть включены в тест сьют, так как это заняло бы у нас очень много времени и в конце концов не стоило бы нам таких усилий. Если каждый тестировщик сядет и тщательно продумает все сценарии и если отдать эту фитчу для тестирования другому тестировщику, то он как правило находит еще кучу возможных сценариев и кейсов, которые можно включить. Поэтому в тестировании ПО принято проанализировать продукт или новую фитчу после чего сфокусировать усилия в тестировании на более рисковые и приоритетные случаи и участки нашего продукта.

2) Скопление багов. Если брать продукт и разбить его по модулям, то в процессе тестирования можно заметить что основная часть багов лежит в одном или нескольких модулях продукта, следовательно можно отметить эффект скопления багов. Как правило такое можно наблюдать в совершенно разных продуктах. Для того чтобы эффективно протестировать наш продукт, следует распределить свои усилия в тестировании по реальной плотности багов в модулях продукта, если же тестируем впервые, то пропорционально ожидаемой плотности. Со временем тенденция с накоплением багов может изменятся от модуля к модулю. Это следует отслеживать и перераспределять усилия в дальнейшем тестировании.

3) Эффективность раннего тестирования. Очень важно чтобы тестирование начиналось как можно раньше и предугадывались возможные ошибки, которые может совершить разработчик.

Перед тем как разрабатывать и тестировать тот или иной продукт, следует выяснить всю специфику, возможные конфликты в спецификации, невозможность новой фитчи взаимодействовать с другим модулем, убедиться, что тестировщик, разработчик и продакт овнер одинаково правильно понимают, как это будет реализовано. Запомните, чем раньше найдутся баги, тем дешевле их исправить.

4) Парадокс пестицида. Если, написав тест кейсы, прогонять их много, много раз, то в конечном счете эти кейсы не помогут нам найти новые баги. Поэтому существует практика в тестировании, когда пересматривают и модернизируют тест кейсы чтобы словить какие-то новые баги. Тест кейсы могут усложняться, быть разносторонними так чтобы охватить все компоненты, модули нашего продукта, что в свою очередь поможет нам найти больше интересных и новых багов.

5) Тестирование зависит от нашего продукта. Существует много программ, продуктов и к каждой из них следует подходить индивидуально в плане тестирования. В каких-то больше усилий в тестировании нужно на безопасность, в каких-то на юзабилити. Поэтому не стоит грести все продукты под одну гребенку и тестировать по какому-то одному шаблону.

6) Тестирование показывает наличие багов в продукте, но ни как не их отсутствие. Многие думают, что, если новый функционал прошел стадию тестирования, значит все — багов уже нет. В этом и заключается ошибочное суждение. Тестирование лишь снижает вероятность наличия багов в продукте. Следовательно, в процессе тестирования много багов может быть пропущено и это не будет означать что если продукт прошел тестирование, то сейчас этот продукт на 100% работает корректно.

7) Продукт хорошо протестирован и вроде бы нет очевидных багов, то это хороший продукт. Иногда тестрируя и выискивая функциональные баги, мы забываем посмотреть с другой стороны и спросить, а нужно ли это пользователю. Если эта фитча не соответствует ожиданиям пользователя и его потребностям, то какой бы качественный наш продукт не был — это уже не так важно.

## 1.2 Методы тестирования

Целью применения различных методологий тестирования в процессе тестирования является обеспечение того, чтобы приложение могло успешно работать и работать, как ожидается, в различных средах и на разных платформах.

Функциональное тестирование включает проверку, все ли бизнес-требования выполняются приложением без каких-либо ошибок. Он включает в себя все возможные типы тестов, чтобы проверить, ведет ли каждая часть приложения так, как ожидается, в соответствии с требованиями бизнеса или клиента.

Методы функционального тестирования в следующем порядке перечислены ниже:

* Модульное тестирование
* Интеграционное тестирование
* Тестирование системы
* Приемочное тестирование

Нефункциональное тестирование выполняется для проверки производительности, удобства использования, надежности, совместимости приложения. Это включает:

* Тестирование производительности
* Тестирование безопасности
* Юзабилити-тестирование
* Тестирование на совместимость

Ключом к разработке и выпуску высококачественного приложения, которое легко воспринимается конечными пользователями, является создание великолепной среды тестирования, которая может выполнять как функциональное, так и нефункциональное тестирование.

1. Модульное тестирование

Единица – самая маленькая часть приложения, которую можно протестировать. Целью модульного тестирования является проверка каждого модуля, чтобы увидеть, был ли он разработан в соответствии с требованиями. Единицей может быть отдельная программа, функция, метод и т. Д.

Модульное тестирование часто выполняется разработчиками. Разработчики выполняют модульное тестирование, выполняя кусок кода перед тем, как передать его группе тестирования.

Модульное тестирование является очень важной частью процесса тестирования, потому что, если модульное тестирование выполнено идеально, дефекты выявляются на ранней стадии, и время для устранения этих ошибок сокращается.

2. Интеграционное тестирование

Интеграционное тестирование – тип тестирования, при котором отдельные блоки группируются и тестируются. Этот тип тестирования проводится для выявления любых видов дефектов во взаимодействии между интегрированными подразделениями или группами.

Все модули объединяются вместе после этапа модульного тестирования для создания компонентов, которые могут выполнять некоторые конкретные задачи. Затем на этих созданных компонентах выполняется интеграционное тестирование, чтобы проверить, ведут ли компоненты себя так, как ожидается, и взаимодействия между блоками бесперебойны. Интеграционное тестирование может выполняться как разработчиками, так и тестерами, либо вручную, либо с использованием автоматизации.

3. Системное тестирование

Системное тестирование выполняется, когда все модули разработаны и интегрированы, чтобы сформировать целостную систему, выполняющую задачу. Системное тестирование проверяет, что система соответствует ее требованиям и работает как ожидалось. Эта полностью интегрированная система может быть определенным интерфейсом или экраном, подобным окну входа в систему.

Этап тестирования системы выполняется группой тестирования, которая тестирует сквозную функциональность приложения, прежде чем приложение кода будет готово к работе.

4. Приемочные испытания

Этап приемочного тестирования – последний этап процесса функционального тестирования, когда приложение проверяется на готовность к доставке клиенту. Этот этап гарантирует, что приложение соответствует всем бизнес-требованиям и критериям и готово к доставке. Приложение тестируется как тестировщиками из компании, так и за ее пределами (бета-тестеры).

Этап бета-тестирования является наиболее важным этапом, когда компания проверяет продукт на реальных конечных пользователях и устраняет любые обнаруженные дефекты.

5. Тестирование производительности

Тестирование производительности – метод тестирования, используемый для проверки того, будет ли приложение вести себя должным образом при большой нагрузке и при других условиях. Тестирование производительности проводится для проверки скорости, масштабируемости и стабильности приложения. Тестирование производительности является основным из 4 типов:

* Нагрузочное тестирование
* Стресс-тестирование
* Тест на выносливость
* Спайк тестирование

6. Тестирование безопасности

Тестирование безопасности – тип тестирования, выполняемый для гарантии того, что данные в приложении безопасны и хорошо защищены. Команда тестирования намеренно пытается найти пути или лазейки, чтобы проникнуть в систему, чтобы получить несанкционированный доступ, который может привести к потере информации или угрозе безопасности. Некоторые из методов тестирования для проверки безопасности:

* целостность
* конфиденциальность
* Аутентификация
* авторизация
* Доступность
* Неотрекаемость

7. Юзабилити-тестирование

Юзабилити-тестирование – методика тестирования, позволяющая конечным пользователям легко использовать приложение. Юзабилити-тестирование проводится с реальными пользователями, чтобы увидеть, насколько легко им выполнить задачу в приложении.

8. Тестирование на совместимость

Тестирование совместимости проводится, чтобы увидеть, как приложение будет работать в разных средах, таких как различное оборудование, операционные системы, мобильные устройства, платформы, браузеры или условия сети. Цель этого тестирования - убедиться, что приложение не зависит от платформы и выполняет ожидаемые функции в различных средах.

Важность методологий тестирования

Методологии тестирования программного обеспечения важны для выявления ошибок и ошибок в приложении, которые были допущены при разработке приложения.

Тестирование приложения гарантирует, что надежность компании поддерживается в сознании пользователей, и они удовлетворены приложением.

Тестирование повышает качество продукта, устраняя любые дефекты в приложении. Продукты лучшего качества поставляются, чтобы повысить доверие пользователей к компании.

Высококачественное тестирование обеспечивает эффективную и бездефектную работу приложения.

Проверка качества гарантирует, что приложение не приведет к каким-либо сбоям, а дефекты будут выявлены как можно раньше. Дефекты, выявленные на более поздних этапах, очень дороги для исправления и могут нанести ущерб репутации компании, если будут обнаружены на производстве.

# 2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ РАСЧЕТ СТОИМОСТИ УСЛУГ, ПРЕДОСТАВЛЯЕМЫХ КЛИЕНТАМ

## 2.1. Разработка тестовой документации (тест-дизайн)

Документ описывает методы и подходы к тестированию, которые будут использоваться тестировщиками отдела тестирования, для тестирования приложения. План тестирования может использоваться как тестировщиками, так и менеджерами, разработчиками. Объект тестирования — это деятельность, направленная на проверку работоспособности функций ИС «Учёт выдачи и возврата книг» приложения «BookCheck».

Целью тестирования приложения «BookCheck» является проверка корректной работы.

Итогом процесса тестирования будут следующие материалы:

1. Определить существующую информацию о проекте и программных компонентах, подлежащих тестированию.
2. Описать стратегии тестирования, которые будут использоваться.
3. Определить необходимые ресурсы для проведения тестирования.
4. Привести результаты тестирования.

Результаты будут отправлены заказчику в виде отчетов.

Условия для тестирования. Приложение должно удовлетворять потребность пользователя в активностях, связанных с просмотром каталога услуг, делать заказы, пользоваться обратной связью, рассчитывать стоимость.

Стратегия процесса тестирования. Приведенный ниже план тестирования является формальным, так как для построения развернутого плана необходимо понимание текущего состояния проекта.

Основными задачами тестирования являются:

* проведение функционального тестирования каждого модуля и компонента системы для обеспечения его соответствия функциональным требованиям;
* тестирование данных и целостности базы данных.

Виды тестирования. Для решения указанных выше задач тестирования будут использоваться следующие виды тестирования.

1. Тестирование данных и целостности базы данных

Базы данных должны тестироваться как отдельные системы внутри ИС «Учёт выдачи и возврата книг». Эти системы должны тестироваться отдельно от приложений (таких как интерфейс доступа к данным).

Необходимо провести дополнительное исследование СУБД на тему того, какие инструменты/техники существуют для выполнения нижеописанного тестирования.

1.1 Цель тестирования

Убедится в том, что методы доступа к данным работают правильно и без нарушения целостности БД.

1.2 Способы

* Вызвать каждый метод доступа к БД, предоставляя правильные и не правильные данные (или запросы к данным).
* Исследовать БД на предмет корректного заполнения ее данными, корректной обработки событий

1.3 Критерий завершенности

Все методы и процедуры БД функционируют так, как им положено и без нарушения целостности самой БД.

1.4 Особые замечания

* При тестировании может понадобиться среда разработки СУБД или драйвера для корректного подключения к базам данных.
* Процедуры должны вызываться вручную.
* Для повышения видимости неприемлемых событий БД необходимо использовать небольшие БД или БД с ограниченным количеством записей.

2. Функциональное тестирование

1.1 Цель тестирования

Функциональное тестирование состоит в том, чтобы убедиться, что весь программный продукт работает в соответствии с требованиями, и в приложении не появляется существенных ошибок.

1.2 Способы

* Авторизация
* Авторизация пользователя
* Редактирование учетной записи
* Оформление выдачи и возврата книги
* Возврат в шагах при оформлении
* Указание даты возврата
* Заполнение контактных данных читателя
* Отправка запроса в БД

1.3 Критерий завершенности

Программный продукт должен пройти все запланированные тесты.

1.4 Особые замечания

Ожидаемые результаты возникают при использовании достоверных данных.

Соответствующие сообщения об ошибках или предупреждения отображаются, когда используются неверные данные.

Подготовлено тестовое окружение, приложение готово к тестированию на тестовой площадке.

Не будет проведено нагрузочное и тестирование безопасности в виду отсутствия необходимых ресурсов.

Отчеты об ошибках создаются для того, чтобы предоставить команде разработчиков и руководителю проекта исчерпывающую информацию об обнаруженных ошибках. Они должны быть полезны при определении причин ошибок и их исправлении.

Продукт должен работать в соответствии с требованиями и техническим заданием. Продукт не должен содержать критических и блокирующих дефектов в окончательной версии проекта.

## 2.2. Разработка тестовых сценариев

Unit test – блок кода (обычно метод), который вызывает тестируемый блок кода и проверяет его правильность работы. Если результат Unit – теста не совпадает с ожидаемым результатом, тест считается не пройденным.

Модульное тестирование (Unit testing) – тестирование каждой атомарной функции приложения отдельно, с использованием объектов искусственно смоделированной среды.

Создадим новый проект – библиотека классов (Class Library), и назовем его MyCalcLib (рисунок 1).

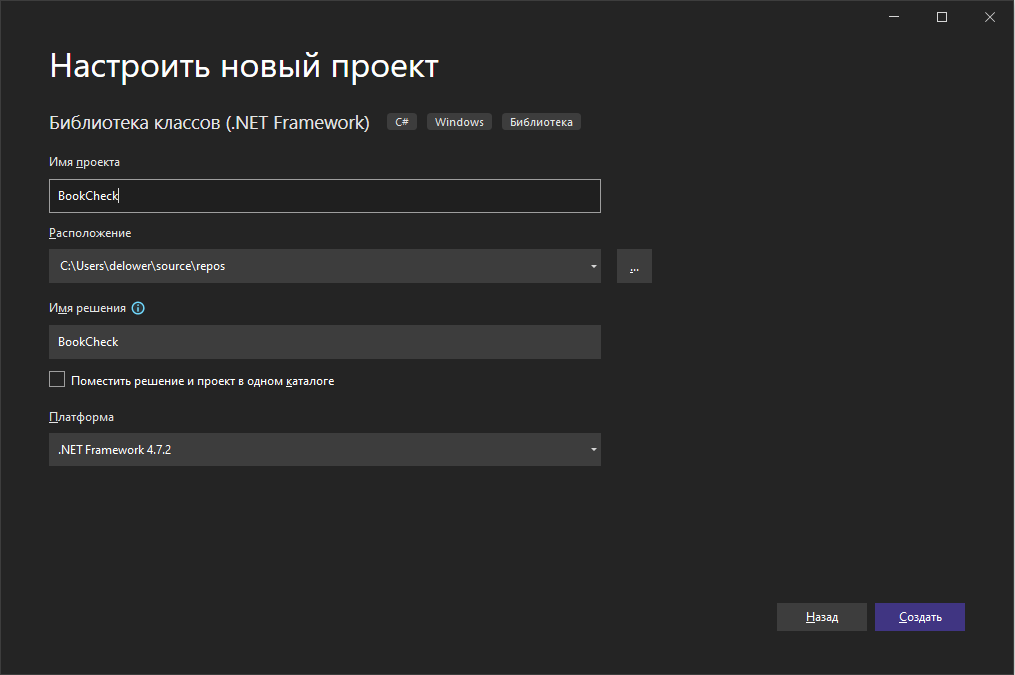


Рисунок 1 – Создание проекта

Переименуем созданный класс из Class1 в loginCheck.

Далее добавим открытый метод, который будет возвращать string, с названием Answer в качестве параметра он принимает string name и string sername, и этом методе возвращаем name + sername. Текст кода предоставлен на рисунке ниже:

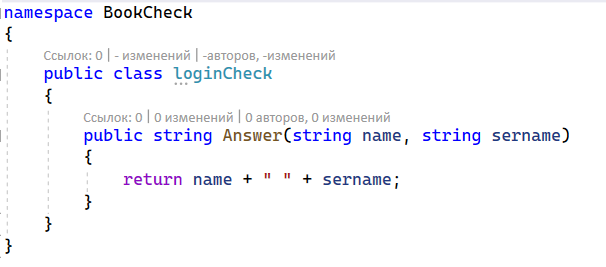


Рисунок 2 – Код метода

Для этого метода нам необходимо создать Unit Test. Правой кнопкой

мыши кликаем в обозреватели решений по решению BookCheck, выбираем

добавить новый проект (рисунок 3). Затем в списке проектов переключаемся на вкладку Test, далее в правой части необходимо указать тип теста: модульный тест, веб-тест или тест пользовательского интерфейса, в данном случае выбираем проект модульного теста (Unit Test Project) и назовем его CheckLogin (рисунок 4).

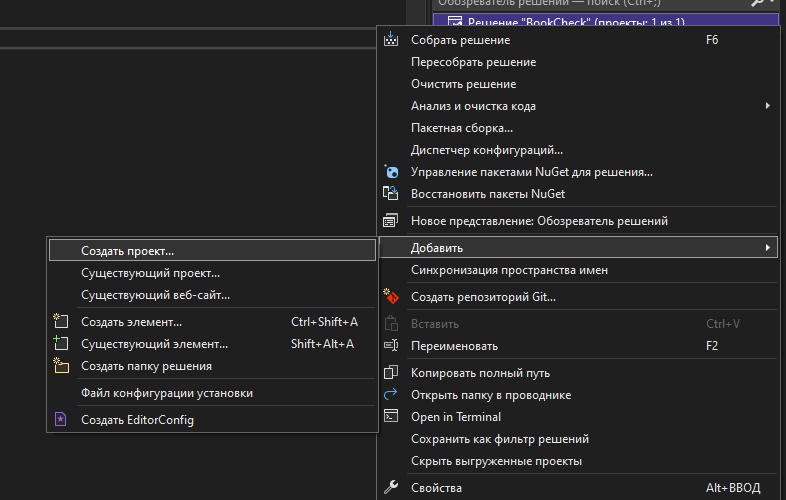
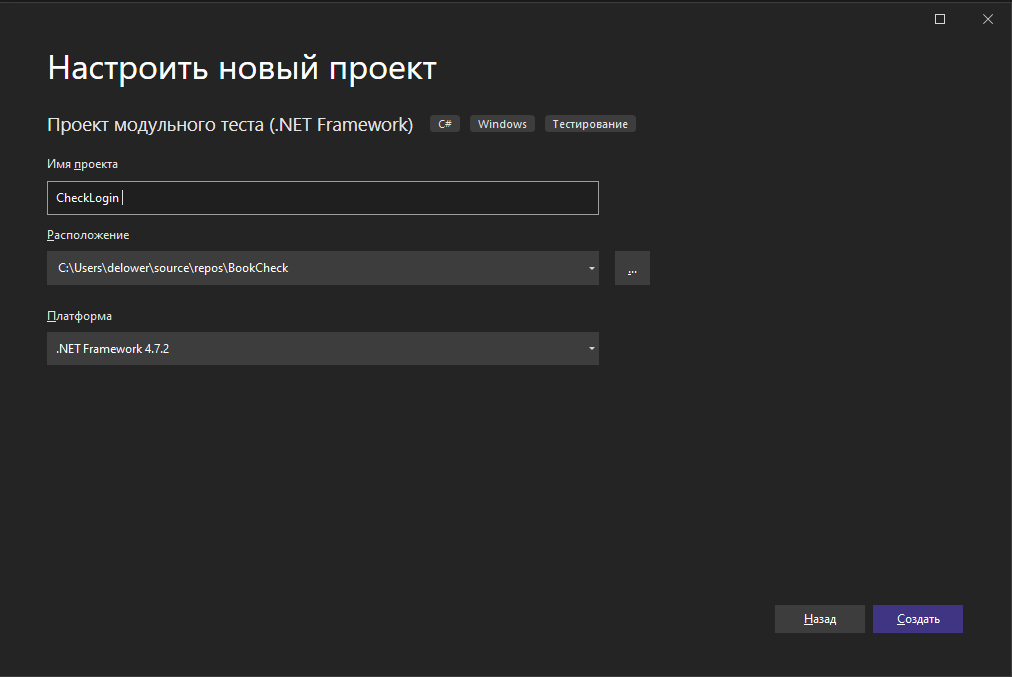


Рисунок 3 – Добавление нового проекта

Рисунок 4 – Окно добавление нового проекта

Переименуем в обозреватели решений UnitTest1 в chkLogin.

Unit – тестами являются методы, которые помечены атрибутом [TestMethod], Unit – тесты собираются в классы, которые помечены атрибутом [TestClass]. Текст кода предоставлен на рисунке ниже:

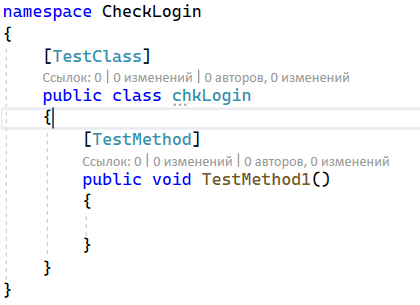


Рисунок 5 – Текст кода

Чтобы использовать TestClass и TestMethod атрибуты необходимо подключить пространство имен Microsoft.VisualStudio.TestTools.UnitTesting.

Для того чтобы мы имели возможность работать с классом loginCheck.cs, который находится в отдельной сборке, необходимо эту сборку подключить, для этого правой кнопкой мыши кликаем в обозреватели решений по папке References в BookCheck.Tests, и в контекстном меню выбираем добавить ссылку.

Выбираем проект BookCheck из текущего решения. Теперь эта сборка подключена.

Теперь необходимо написать Unit – тест, который будет проверять правильность работы метода Answer, для этого вначале переименуем имя метода из TestMethod1 в chkLoginTest (это метод на вход будет принимать два значения name и sername, результат, который будет возвращен этим методом будет равен expected), это необходимо для того чтобы четко понимать и отображать что именно мы тестируем. Далее мы делаем три операции arrange (здесь создаются переменные), act (здесь выполняются действия над системой) и assert (проверка, что операции выполнились успешно и Unit – тест завершен правильно). Текст кода представлен на рисунке ниже:

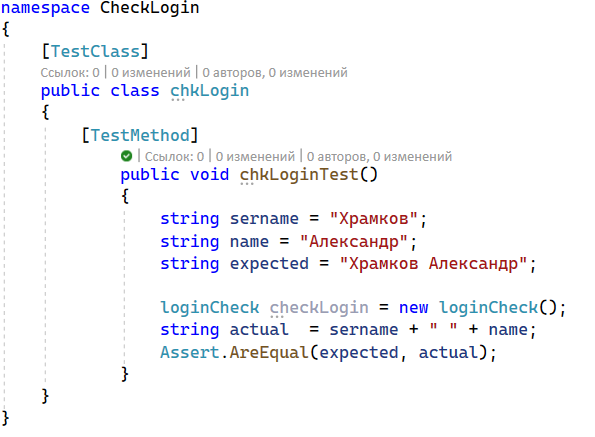


Рисунок 6 – Текст кода

Если в результате выполнения фрагмента кода приведенного выше значения expected и actual будут равны, то в таком случае мы получим результат true и Unit – тест – закончится успешно, если же они не будут равны, то мы получим ошибку.

Для запуска теста в обозреватели теста (Test Explorer), для того чтобы вызвать окно обозревателя решений кликаем ТЕСТ – окна – обозреватель тестов, кликом по необходимому тесту вызвать контекстное меню и в контекстном меню кликнуть по выполнить выбранные тесты или же в верхней панели в закладке тест пройти по вкладке выполнить и в нем выбрать необходимый вариант. Если тесты отсутствуют в обозреватели решений, необходимо выполнить сборку решения, для этого в верхней панели выбираем СБОРКА (Build) – Собрать решение.

Когда тест завершился если он завершился удачно рядом с тестом должна появиться зеленая иконка и также в нижней части обозревателя теста не должно выводиться не каких ошибок. Также после успешного завершения теста можно посмотреть время его исполнения (в нашем случае это 51 мс).

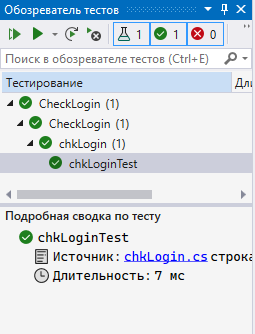


Рисунок 7 – Результат теста

Если мы выполним изменения в методе Answer, и они приведут к тому, что появятся ошибки, то после запуска Unit – теста, мы увидим какая произошла ошибка и в каком методе. Красная иконка свидетельствует о том, что в тесте произошел сбой.

# ЗАКЛЮЧЕНИЕ

В первой части курсовой работы были проанализированы методы тестирования. Тестирование – это процесс анализа пункта требований к ПО с целью фиксации различий между существующим состоянием ПО и требуемым (что свидетельствует о проявлении ошибки) при экспериментальной проверке соответствующего пункта требований. Были изучены критерии и принципы тестирования информационной системы. Также, были описаны методы тестирования ИС, такие как метод «черного ящика», метод «белого ящика», метод «серого ящика».

Во второй части курсовой работы на основе изученных методов тестирования была разработана и подробно описана тестовая документация (тест-дизайн) информационной системы, а также был разработан тестовый сценарий для информационной системы «Учёт выдачи и возврата книг». В целом это составляет тестирование ИС.

Практической частью курсовой работы было тестирование фрагмента информационной системы «Учёт выдачи и возврата книг».

Подробно описаны и проиллюстрированы функционал и структура самого модуля, а также проведено тестирование готового программного продукта в программе MS Visual Studio 2022. В последнюю очередь была составлена сопутствующая документация.

Таким образом, задачи, сформулированные во введении, решены, а цель достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 34.601 – 90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.

2. ГОСТ 34.602 - 2020. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.

3. ГОСТ 19.201 - 78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.

4. ГОСТ 19.202 - 78 ЕСПД. Спецификация. Требования к содержанию и оформлению.

5. ГОСТ Р ИСО/МЭК 12207. Процессы жизненного цикла программных средств.

6. Золотов С.Ю. Проектирование информационных систем: Учебно-методическое пособие. Методические рекомендации для выполнения курсового проекта, лабораторных работ и практических занятий по дисциплине «Проектирование информационных систем» - Томск: ТУСУР, 2013. - 34 с.

7. Шнайдер, Роберт Microsoft SQL Server 6.5. Проектирование высокопроизводительных баз данных; М.: Лори, 2010. - 361 c

8. Петкович, Душан Microsoft SQL Server 2012. Руководство для начинающих / Душан Петкович. - М.: БХВ-Петербург, 2012. - 460 c.

9. Тейлор, Аллен SQL для чайников / Аллен Тейлор. - М.: Вильямс, 2014. - 416 c.

10. Браст, Э.Дж. Разработка приложений на основе Microsoft SQL Server 2008 / Э.Дж. Браст. - М.: Русская Редакция, 2010. - 751 c.

11. Хетагуров, Я. А. Проектирование автоматизированных систем обработки информации и управления (АСОИУ). Учебник / Я.А. Хетагуров. - М.: Бином. Лаборатория знаний, 2015. - 240 c.

12. Кристофер, Д. Маннинг Введение в информационный поиск / Кристофер Д. Маннинг, ПрабхакарРагхаван ,ХайнрихШютце. - М.: Вильямс, 2014. - 528 c.

13. Бишоп Дж. С# в кратком изложении; Бином. Лаборатория знаний - М., 2015. - 234 c.

14. ГриффитсИэн Программирование на C# 5.0; Эксмо - М., 2014. - 580 c.

15. Гуриков С. Р. Введение в программирование на языке Visual C#; ИЛ - Москва, 2013. - 448 c.

16. Шилдт Герберт C# 4.0. Полное руководство; Вильямс - М., 2015. - 291 c.

17. Эндрю Троелсен Язык программирования C# 5.0 и платформа .NET 4.5; Диалектика / Вильямс - М., 2015. - 126 c.