Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

**КУРСОВАЯ РАБОТА**

**ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ КНИЖНОГО МАГАЗИНА ООО «БУКВОГЛОТ»**

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

**МДК 05.03 «ТЕСТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ»**

**09.02.07 «ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ»**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  | **/** | Чулюкина Т.О. |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 г. |  |  |  |
| Руководитель |  | **/** | *Н.К. Коровина* |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 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

*Чулюкиной Татьяны Олеговны*

*Фамилия Имя отчество студента*

|  |  |
| --- | --- |
| Тема курсовой работы : | «Книжный магазин» |

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 книжного магазина

2.1. Разработка тестовой документации (тест-дизайн)

2.2. Разработка тестовых сценариев

Заключение

Список использованных источников

Приложения

Дата выдачи задания: « » 202 г.

Дата сдачи работы на отделение: « » 202 г.

Руководитель курсового(ой) проекта(работы) \_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись расшифровка подписи

Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

**КАЛЕНДАРНЫЙ ПЛАН**

выполнения курсовой работы

Студентом 3 курса ИСП-31 группы Чулюкиной Т. О.

*Фамилия, И.О.*

По теме **Книжный магазин**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  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 |  | **/** | **Чулюкина Т. О.** |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 г. |  |  |  |
| Руководитель |  | **/** | *Н.К. Коровина* |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 г. |  |  |  |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc100908991)

[1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ 6](#_Toc100908992)

[1.1 Критерии и принципы тестирования 6](#_Toc100908993)

[1.2 Методы тестирования 7](#_Toc100908994)

[2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ КНИЖНОГО МАГАЗИНА 13](#_Toc100908995)

[2.1. Разработка тестовой документации (тест-дизайн) 13](#_Toc100908996)

[2.2. Разработка тестовых сценариев 15](#_Toc100908997)

[ЗАКЛЮЧЕНИЕ 19](#_Toc100908998)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc100908999)

[ПРИЛОЖЕНИЯ 20](#_Toc100909000)

# ВВЕДЕНИЕ

Процесс автоматизации начался намного раньше, чем нам могло бы казаться, автоматизация на самом деле появилась практически сразу же с возникновением производства, а само по себе производство существует уже так давно, что точно никто и не скажет. Высокая экономическая эффективность, технологическая целесообразность и часто эксплуатационная необходимость способствовали широкому распространению автоматизации в промышленности, на транспорте, в торговле и различных сферах обслуживания.

Автоматизированная информационная система данных осуществляет хранение, редактирование, ведение, создание диаграмм и отчетов.

Актуальность темы: определяется тем, что в связи с моральным и физическим устареванием платформы, форматов невозможно использовать существующую автоматизированную информационную систему, которая замедляет работу, частыми техническими сбоями в системе и определяется актуальность разработки информационной системы книжного магазина.

Все выше изложенное в целом на теоретико-методологическом уровне определило проблему настоящего исследования: разработка программного модуля ИС «Книжный магазин».

1. Анализ предметной области.
2. Разработка функциональной модели.
3. Разработка логической и физической структуры.
4. Разработка информационной системы «Книжный магазин».
5. Разработка интерфейса ИС.
6. Разработка руководства пользователя.
7. Тестирование разработанного ПП.

Методы исследования: Для разработки информационной системы будут применяться методы анализ деятельности предприятия с выявлением его функции, а также функциональное моделирование системы.

# 1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ

## 1.1 Критерии и принципы тестирования

Структурные критерии используют информацию о структуре программы, что предполагает знание исходного текста программы или спецификации программы в виде потокового графа управления. Структурные критерии базируются на основных элементах графа управления - операторах, ветвях и путях.

* Условие критерия тестирования команд (критерий С0) - набор тестов в совокупности должен обеспечить прохождение каждой команды не менее одного раза.
* Условие критерия тестирования ветвей (критерий С1) - набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза.
* Условие критерия тестирования путей (критерий С2) - набор тестов в совокупности должен обеспечить прохождение каждого пути не менее 1 раз.

Принцип 1 — Тестирование демонстрирует наличие дефектов (Testing shows presence of defects). Тестирование может показать, что дефекты присутствуют, но не может доказать, что их нет. Тестирование снижает вероятность наличия дефектов, находящихся в программном обеспечении, но не гарантирует их отсутствия.

Принцип 2 — Исчерпывающее тестирование недостижимо (Exhaustive testing is impossible). Полное тестирование с использованием всех комбинаций вводов и предусловий физически невыполнимо, за исключением тривиальных случаев. Вместо исчерпывающего тестирования должны использоваться анализ рисков и расстановка приоритетов, чтобы более точно сфокусировать усилия по тестированию.

Принцип 3 — Раннее тестирование (Early testing). Чтобы найти дефекты как можно раньше, активности по тестированию должны быть начаты как можно раньше в жизненном цикле разработки ПО или системы, и должны быть сфокусированы на определенных целях.

Принцип 4 — Скопление дефектов (Defects clustering). Разные модули системы могут содержать разное количество дефектов – то есть плотность скопления дефектов в разных частях кода может отличаться. Усилия по тестированию должны распределяться пропорционально фактической плотности дефектов. В основном, большую часть критических дефектов находят в ограниченном количестве модулей.

Принцип 5 — Парадокс пестицида (Pesticide paradox). Если одни и те же тесты будут прогоняться много раз, в конечном счете этот набор тестовых сценариев больше не будет находить новых дефектов. Чтобы преодолеть этот «парадокс пестицида», тестовые сценарии должны регулярно рецензироваться и корректироваться, новые тесты должны быть разносторонними, чтобы охватить все компоненты программного обеспечения, или системы, и найти как можно больше дефектов.

Принцип 6 — Тестирование зависит от контекста (Testing is concept depending). Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем новостной портал.

Принцип 7 — Заблуждение об отсутствии ошибок (Absence-of-errors fallacy). Отсутствие найденных дефектов при тестировании не всегда означает готовность продукта к релизу. Система должна быть удобна пользователю в использовании и удовлетворять его ожиданиям и потребностям.

## 1.2 Методы тестирования

Все виды тестирования программного обеспечения, в зависимости от преследуемых целей, можно условно разделить на следующие группы:

* функциональные;
* нефункциональные;
* связанные с изменениями.

**Функциональные виды тестирования.**

Функциональные тесты базируются на функциях и особенностях, а также на взаимодействии с другими системами и могут быть представлены на всех уровнях тестирования: компонентном или модульном (Component/Unit testing), интеграционном (Integration testing), системном (System testing), приемочном (Acceptance testing). Функциональные виды тестирования рассматривают внешнее поведение системы. Далее перечислены одни из самых распространенных видов функциональных тестов.

Функциональное тестирование рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.

**1. Функциональные тесты** основываются на функциях, выполняемых системой, и могут проводиться на всех уровнях тестирования (компонентном, интеграционном, системном, приемочном). Как правило, эти функции описываются в требованиях, функциональных спецификациях или в виде случаев использования системы (use cases).

Тестирование функциональности может проводиться в двух аспектах:

* Требования.
* Бизнес-процессы.

Тестирование в аспекте «требования» использует спецификацию функциональных требований к системе, как основу для дизайна тестовых случаев (Test Cases). В этом случае необходимо сделать список того, что будет тестироваться, а что нет, приоритезировать требования на основе рисков (если это не сделано в документе с требованиями), а на основе этого приоритезировать тестовые сценарии (test cases). Это позволит сфокусироваться и не упустить при тестировании наиболее важный функционал.

Тестирование в аспекте «бизнес-процессы» использует знание бизнес-процессов, которые описывают сценарии ежедневного использования системы. В этом аспекте тестовые сценарии (test scripts), как правило, основываются на случаях использования системы (use cases).

**2. Тестирование безопасности (Security and Access Control Testing)**

Тестирование безопасности - это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Общая стратегия безопасности основывается на трех основных принципах:

* Конфиденциальность.
* Целостность.
* Доступность.

Конфиденциальность - это сокрытие определенных ресурсов или информации. Под конфиденциальностью можно понимать ограничение доступа к ресурсу некоторой категории пользователей или, другими словами, при каких условиях пользователь авторизован получить доступ к данному ресурсу.

Существует два основных критерия при определении понятия целостности:

* Доверие. Ожидается, что ресурс будет изменен только соответствующим способом определенной группой пользователей.
* Повреждение и восстановление. В случае, когда данные повреждаются или неправильно меняются авторизованным или не авторизованным пользователем, Вы должны определить, насколько важной является процедура восстановления данных.
* Доступность. Доступность представляет собой требования о том, что ресурсы должны быть доступны авторизованному пользователю, внутреннему объекту или устройству. Как правило, чем более критичен ресурс, тем выше уровень доступности должен быть.

**3.** **Тестирование взаимодействия или Interoperability Testing**

Тестирование взаимодействия (Interoperability Testing) – это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя тестирование совместимости (compatibility testing) и интеграционное тестирование (integration testing).

Программное обеспечение с хорошими характеристиками взаимодействия может быть легко интегрировано с другими системами, не требуя каких-либо серьезных модификаций. В этом случае, количество изменений и время, требуемое на их выполнение, могут быть использованы для измерения возможности взаимодействия.

**Нефункциональные виды тестирования.**

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. В целом, это тестирование того, как система работает.

1. **Все виды тестирования производительности**

**Тестирование производительности (Performance testing).**

Задачей тестирования производительности является определение масштабируемости приложения под нагрузкой, при этом происходит:

Измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций.

Определение количества пользователей, одновременно работающих с приложением.

Определение границ приемлемой производительности при увеличении нагрузки (при увеличении интенсивности выполнения этих операций).

Исследование производительности на высоких, предельных, стрессовых нагрузках.

**Стрессовое тестирование (Stress Testing).**

Стрессовое тестирование позволяет проверить, насколько приложение и система в целом работоспособны в условиях стресса, а также оценить способность системы к регенерации, т.е. к возвращению к нормальному состоянию, после прекращения воздействия стресса. Стрессом, в данном контексте, может быть повышение интенсивности выполнения операций до очень высоких значений или аварийное изменение конфигурации сервера. Также, одной из задач при стрессовом тестировании может быть оценка деградации производительности. Таким образом, цели стрессового тестирования могут пересекаться с целями тестирования производительности.

**Объемное тестирование (Volume Testing).**

Задачей объемного тестирования является получение оценки производительности при увеличении объемов данных в базе данных приложения, при этом происходит:

Измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций.

Может производиться определение количества пользователей, одновременно работающих с приложением.

**Тестирование стабильности или надежности( Stability / Reliability Testing).**

Задачей тестирования стабильности (надежности) является проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки. Время выполнения операций может играть в данном виде тестирования второстепенную роль. При этом на первое место выходит отсутствие утечек памяти, перезапусков серверов под нагрузкой и другие аспекты, влияющие именно на стабильность работы.

В англоязычной терминологии вы можете так же найти еще один вид тестирования - Load Testing - тестирование реакции системы на изменение нагрузки (в пределе допустимого). Нам показалось, что Load и Performance преследуют все же одну и ту же цель: проверка производительности (времени отклика) на разных нагрузках. Собственно поэтому мы и не стали разделять их. В то же время кто то может разделить. Главное все-таки понимать цели того или иного вида тестирования и постараться их достигнуть.

**2. Тестирование установки (Installation Testing).**

Тестирование установки направленно на проверку успешной инсталляции и настройки, а также на обновление или удаление программного обеспечения.

В настоящий момент, наиболее распространена установка ПО при помощи инсталляторов (специальных программ, которые сами по себе так же требуют надлежащего тестирования, описание которого рассмотрено в разделе "Особенности тестирования инсталляторов").

В реальных условиях инсталляторов может не быть. В этом случае придется самостоятельно выполнять установку программного обеспечения, используя документацию в виде инструкций или "read me" файлов, шаг за шагом описывающих все необходимые действия и проверки.

В распределенных системах, где приложение разворачивается на уже работающем окружении, простого набора инструкций может быть мало. Для этого часто пишется план установки (Deployment Plan), включающий не только шаги по инсталляции приложения, но и шаги отката (roll-back) к предыдущей версии (в случае неудачи). Сам по себе план установки также должен пройти процедуру тестирования для избежания проблем при выдаче в реальную эксплуатацию. Особенно это актуально, если установка выполняется на системы, где каждая минута простоя - это потеря репутации и большого количества средств, например: банки, финансовые компании или даже баннерные сети. Поэтому тестирование установки можно назвать одной из важнейших задач по обеспечению качества программного обеспечения.

**3. Тестирование удобства пользования (Usability Testing).**

Иногда мы сталкиваемся с непонятными или нелогичными приложениями, многие функции и способы использования которых часто не очевидны. После такой работы редко возникает желание использовать приложение снова, и мы ищем более удобные аналоги. Для того, чтобы приложение было популярным, ему мало быть функциональным – оно должно быть еще и удобным. Если задуматься, интуитивно понятные приложения экономят нервы пользователям и затраты работодателя на обучение.

# 2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ УЧЁТ КОНТИНГЕНТА УЧАЩИХСЯ УЧЕБНОГО ЗАВЕДЕНИЯ

## 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 приложения ИС книжного магазина «Буквоглот».

Целью тестирования приложения ИС книжного магазина «Буквоглот» 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 работ по тестированию.
4. Привести результаты тестирования.

Результаты будут отправлены заказчику в виде отчетов.

Условия для тестирования. Приложение должно удовлетворять потребность пользователя в активностях, связанных с вводом, редактированием и удалением данных об книгах, о сотрудниках, об авторах, об издательствах, о жанрах, о клиентах; формированием данных к продаже.

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 книжного магазина «Буквоглот». 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

**Авторизация**

Авторизация пользователя

**Просмотр информации**

Просмотр информации о книгах

Просмотр информации о заказах

Просмотр информации о сотрудниках

Просмотр информации о клиентах

Просмотр информации о поставках

Выход

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

После установки базы данных можно приступить к ее использованию в тестах. В большинстве простых случаев в наборе тестов есть отдельная база данных, которая совместно используется несколькими тестами в нескольких классах тестирования, поэтому нам нужна некоторая логика, чтобы убедиться, что база данных создана и заполняется ровно один раз в течение всего времени существования тестового запуска.

При использовании Xunit это можно сделать с помощью [средства](https://xunit.net/docs/shared-context#class-fixture) класса, представляющего базу данных и совместно используемой для нескольких тестовых запусков:

public class TestDatabaseFixture

{

private const string ConnectionString = @"Server=(localdb)\mssqllocaldb;Database=EFTestSample;Trusted\_Connection=True";

private static readonly object \_lock = new();

private static bool \_databaseInitialized;

public TestDatabaseFixture()

{

lock (\_lock)

{

if (!\_databaseInitialized)

{

using (var context = CreateContext())

{

context.Database.EnsureDeleted();

context.Database.EnsureCreated();

context.AddRange(

new Blog { Name = "Blog1", Url = "http://blog1.com" },

new Blog { Name = "Blog2", Url = "http://blog2.com" });

context.SaveChanges();

}

\_databaseInitialized = true;

}

}

}

public BloggingContext CreateContext()

=> new BloggingContext(

new DbContextOptionsBuilder<BloggingContext>()

.UseSqlServer(ConnectionString)

.Options);

}

При создании экземпляра приведенного выше средства используется [EnsureDeleted()](https://docs.microsoft.com/ru-RU/dotnet/api/microsoft.entityframeworkcore.storage.idatabasecreator.ensuredeleted#microsoft-entityframeworkcore-storage-idatabasecreator-ensuredeleted) для удаления базы данных (если она существует из предыдущего запуска), а затем [EnsureCreated()](https://docs.microsoft.com/ru-RU/dotnet/api/microsoft.entityframeworkcore.storage.idatabasecreator.ensurecreated#microsoft-entityframeworkcore-storage-idatabasecreator-ensurecreated) для ее создания с помощью последней конфигурации модели ([см. документацию по этим API](https://docs.microsoft.com/ru-ru/ef/core/managing-schemas/ensure-created)). После создания базы данных средство заменит его некоторыми данными, которые могут использовать наши тесты. Стоит потратить некоторое время на размышления о начальных данных.

Чтобы использовать приспособление в тестовом классе, просто реализуйте IClassFixture его над типом светильника, а xUnit внедряет его в конструктор:

public class BloggingControllerTest : IClassFixture<TestDatabaseFixture>

{

public BloggingControllerTest(TestDatabaseFixture fixture)

=> Fixture = fixture;

public TestDatabaseFixture Fixture { get; }

Теперь класс тестирования имеет Fixture свойство, которое можно использовать тестами для создания полнофункциональный экземпляр контекста:

[Fact]

public void GetBlog()

{

using var context = Fixture.CreateContext();

var controller = new BloggingController(context);

var blog = controller.GetBlog("Blog2").Value;

Assert.Equal("http://blog2.com", blog.Url);

}

Вот здесь можно заметить некоторые блокировки в логике создания светильника выше. Если приспособление используется только в одном тестовом классе, оно гарантированно будет создано ровно один раз с помощью xUnit; Но обычно один и тот же компонент базы данных используется в нескольких тестовых классах.

В приведенном выше примере показан тест только для чтения, который является простым случаем с точки зрения изоляции теста: так как ничего не изменяется, вмешательство теста невозможно. В отличие от этого, тесты, которые изменяют данные, являются более проблематичными, так как они могут вмешиваться друг в друга. Одним из распространенных способов изоляции тестов является перенос теста в транзакцию и откат этой транзакции в конце теста. Так как в базе данных ничего не зафиксировано, другие тесты не видят никаких изменений и интерференции не будут избегать.

Ниже приведен метод контроллера, который добавляет блог в нашу базу данных:

[HttpPost]

public ActionResult AddBlog(string name, string url)

{

\_context.Blogs.Add(new Blog { Name = name, Url = url });

\_context.SaveChanges();

return Ok();

}

Мы можем протестировать этот метод следующим образом:

[Fact]

public void AddBlog()

{

using var context = Fixture.CreateContext();

context.Database.BeginTransaction();

var controller = new BloggingController(context);

controller.AddBlog("Blog3", "http://blog3.com");

context.ChangeTracker.Clear();

var blog = context.Blogs.Single(b => b.Name == "Blog3");

Assert.Equal("http://blog3.com", blog.Url);

}

Некоторые примечания к приведенному выше коду теста:

* Запускаем транзакцию, чтобы убедиться, что приведенные ниже изменения не зафиксированы в базе данных и не вмешиваются в другие тесты. Так как транзакция никогда не фиксируется, она неявно откатывается в конце теста при удалении экземпляра контекста.
* После внесения нужных обновлений очистим средство отслеживания [ChangeTracker.Clear](https://docs.microsoft.com/ru-RU/dotnet/api/microsoft.entityframeworkcore.changetracking.changetracker.clear)изменений экземпляра контекста, чтобы убедиться, что мы фактически загружаем блог из базы данных ниже. Вместо этого можно использовать два экземпляра контекста, но затем необходимо убедиться, что одна и та же транзакция используется обоими экземплярами.

При построении проекта тестирования тесты появляются в обозревателе тестов. Если обозреватель тестов не виден, выберите Тест в меню Visual Studio, Windows, затем обозреватель тестов (или нажмите клавиши CTRL + E, T).

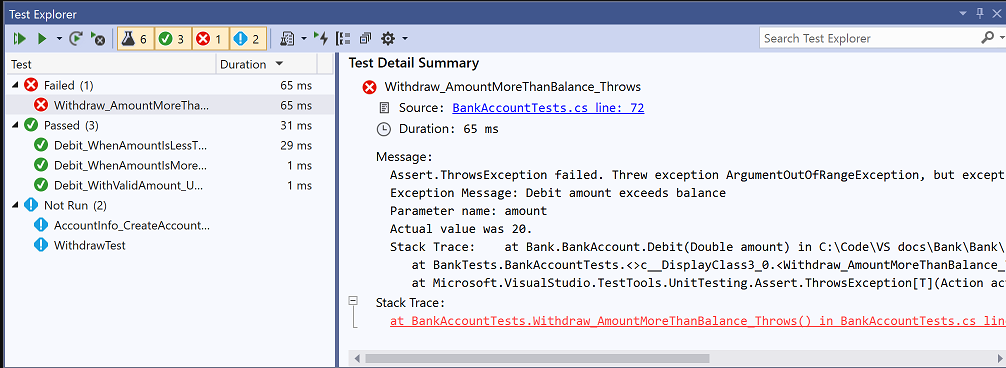


Рисунок 1 - Обозреватель модульных тестов

При выполнении, написании и повторном запуске тестов обозреватель тестов может отображать результаты в группах Неудачные тесты, Пройденные тесты, Пропущенные тесты и Незапущенные тесты. Можно выбирать различные группы по параметрам на панели инструментов.

# ЗАКЛЮЧЕНИЕ

В настоящее время происходит активное развитие информационных технологий. Повышение мощности компьютерной техники требует разработки нового программного обеспечения, способного выполнять разнообразные задачи по обработке информации. Появление нового поколения операционных систем требует разработку совместимых версий программного обеспечения. Специалисты предприятий всего мира уже не могут представить себе работу без вычислительной техники. Автоматизированное рабочее место повышает скорость и качество работы.

# Цель данной курсовой работы — это тестирование разработанного приложения для ИС книжного магазина «Буквоглот». Был проведен анализ методов тестирования:

* критериев и принципов тестирования;
* методов тестирования.

А так же было осуществлено само тестирование информационной системы книжного магазина, была разработана тестовая документация (тест-дизайн) и тестовые сценарии.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 34.201-89 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Виды, комплектность и обозначение документов при создании автоматизированных систем.
2. ГОСТ 34.601-90 ЕСС АСУ. Автоматизированные системы. Стадии создания.
3. ГОСТ 34.602-89 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
4. РД 50-34.698-90 МЕТОДИЧЕСКИЕ УКАЗАНИЯ. ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Автоматизированные системы. Требования к содержанию документов.
5. ГОСТ 21.404-85 Автоматизация технологических процессов. Обозначения условные приборов и средств автоматизации в схемах.
6. ГОСТ 34.603-92 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Виды испытаний автоматизированных систем.
7. ГОСТ 24.701-86 ЕСС АСУ. Надёжность автоматизированных систем управления. Основные положения.
8. ГОСТ 25861-83 Машины вычислительные и системы обработки данных.
9. СНиП 3.05.07-85 Системы автоматизации.
10. СТП 7.3-03-2008 СТАНДАРТ ПРЕДПРИЯТИЯ. Порядок разработки, внедрения, сопровождения и эксплуатации автоматизированных систем управления технологическими процессами.
11. ГОСТ 34.003-90 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Автоматизированные системы. Термины и определения.
12. ГОСТ 24.104-85 ЕСС АСУ. Автоматизированные системы управления. Общие требования.