Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

**КУРСОВАЯ РАБОТА**

**ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «УЧЁТ СОТРУДНИКОВ ПРЕДПРИЯТИЯ»**

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

**МДК 05.03 «ТЕСТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ»**

**09.02.07 «ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ»**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  | **/** | *М.В Немов* |
|  | *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

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Немова Макария Владимировича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия Имя отчество студента*

|  |  |
| --- | --- |
| Тема курсовой работы : | «**Учёт сотрудников предприятия**» |

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

Приложения

Дата выдачи задания: « » 2022 г.

Дата сдачи работы на отделение: « » 2022 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 Немов М.В.

По теме **Учёт сотрудников предприятия**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  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 |  | **/** | *М.В Немов* |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 г. |  |  |  |
| Руководитель |  | **/** | *Н.К. Коровина* |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 г. |  |  |  |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc101788780)

[1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ 6](#_Toc101788781)

[1.1 Критерии и принципы тестирования 6](#_Toc101788782)

[1.2 Методы тестирования 8](#_Toc101788783)

[2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ УЧЁТ СОТРУДНИКОВ ПРЕДПРИЯТИЯ 11](#_Toc101788784)

[2.1. Разработка тестовой документации (тест-дизайн) 11](#_Toc101788785)

[2.2. Разработка тестовых сценариев 14](#_Toc101788786)

[ЗАКЛЮЧЕНИЕ 19](#_Toc101788787)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc101788788)

# ВВЕДЕНИЕ

В настоящее время учёт сотрудников на предприятие и ведение кадрового делопроизводства является обязательным условием эффективного управления персоналом любой компании. Неукоснительного соблюдения требований к документам по учету личного состава требует действующее гражданское, налоговое и трудовое законодательство. Кроме этого, кадровое делопроизводство является немаловажной составляющей трудовых взаимоотношений, оно официально отражает отношения работодателя и наемного работника.

Кадры — это совокупность работников различных профессионально - квалификационных групп, занятых на предприятии и входящих в его списочный состав.

Сегодня множество людей открывают для себя необходимость и важность использования приложения для учета сотрудников предприятия в производстве и организации деятельности. Одним из основных элементов такого механизма является разработка приложения для учета сотрудников предприятия, а также бизнес-процессов, которые не только отражают функциональную и информационную деятельность, но и воздействуют на нее.

Целью курсовых работ является создание информационной системы «Учёт сотрудников предприятия». Для достижения поставленной цели были определены следующие задачи:

* описать предметную область;
* спроектировать программный комплекс;
* выполнить программную реализацию;
* проанализировать качество программного обеспечения.

В данной курсовой работе внимание будет уделено анализу качества программного обеспечения.

# 1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ

## 1.1 Критерии и принципы тестирования

Можно выделить требования к идеальному критерию тестирования:

* критерий должен быть достаточным, т.е. показывать, когда некоторое конечное множество тестов достаточно для тестирования данной программы;
* критерий должен быть полным, т.е. в случае ошибки должен существовать тест из множества тестов, удовлетворяющих критерию, который раскрывает ошибку;
* критерий должен быть надежным, т.е. любые два множества тестов, удовлетворяющих ему, одновременно должны раскрывать или не раскрывать ошибки программы;
* критерий должен быть легко проверяемым, например, вычисляемым на тестах.

Для нетривиальных классов программ в общем случае не существует полного и надежного критерия, зависящего от программ или спецификаций. Поэтому, как правило, стремятся к идеальному общему критерию через реальные частные.

Классы критериев:

* структурные критерии используют информацию о структуре программы (критерии так называемого «белого ящика»);
* функциональные критерии формулируются в описании требований к программному изделию (критерии так называемого «черного ящика»);
* критерии стохастического тестирования формулируются в терминах проверки наличия заданных свойств у тестируемого приложения, средствами проверки некоторой статистической теории;
* мутационные критерии ориентированы на проверку свойств программного изделия на основе подхода Монте-Карло.

Из принципов можно отметить следующее:

Принцип 1. Определение. Для того чтобы протестировать программу, нужно попытаться заставить ее работать неверно.

В силу этого принципа процесс тестирования обретает цель: его единственная задача — найти ошибки, инициируя неудачное выполнение. Любое умозаключение по поводу качества относится к области гарантии качества, но никак не к области тестирования. Это определение также напоминает нам, что тестирование, в отличие от отладки, не связано с исправлением ошибок, — оно связано только лишь с их поиском.

Принцип 2. Тесты или спецификации. Тесты не заменяют спецификации.

Опасность заблуждения, что тесты могут выступать в роли спецификаций, была продемонстрирована целым рядом программных катастроф, которые произошли только потому, что никто не подумал об исключительном случае. Несмотря на то, что в спецификациях тоже могут быть упущены из виду какие-то случаи, по крайней мере, в них предпринимается попытка некого обобщения. В частности, спецификации могут использоваться для генерации тестов, даже автоматически (как в случае тестирования, опирающегося на модели); но обратное без вмешательства человека невозможно.

Принцип 3. Регрессивное тестирование. Любое неудачное выполнение должно порождать тестовый случай, который навсегда становится частью тестового пакета данного проекта.

Этот принцип касается всех ошибок, возникших во время разработки и тестирования. Отсюда вытекает необходимость в инструментарии, позволяющем превращать неудачное исполнение в воспроизводимый тестовый случай, и недавно такой инструментарий появился: Contract-Driven Development, ReCrash, JCrasher.

Принцип 4. Использование предсказаний. Определение успеха или неудачи тестов должно происходить автоматически.

Эта формулировка оставляет открытым вопрос о форме таких предсказаний. Зачастую предсказания специфицируются отдельно. В исследованиях, таких как это, они встроенные, поскольку анализируемое программное обеспечение уже включает в себя контракты, согласно которым тесты используются как предсказания.

Принцип 5. Тестовые случаи, проверяемые вручную и автоматически. Эффективный процесс тестирования должен включать в себя тестовые случаи, проверяемые как вручную, так и автоматически.

Достоинством тестов, выполняемых вручную, является их глубина: они отражают понимание разработчиком имеющегося круга проблем и структуры данных. Преимущество автоматических тестов в их широте: они выполняют проверку большого диапазона значений, в том числе экстремальных, которые люди могут пропустить.

Принцип 6. Эмпирические оценки стратегий тестирования. Желательно оценивать любую стратегию тестирования, однако, какой бы интересной она ни казалась, необходимо прибегать к объективной оценке, используя точные критерии в воспроизводимом процессе тестирования.

Принцип 7. Критерий оценки. Самое важное свойство стратегии тестирования — это число обнаруженных ошибок как функция времени.

Функция обнаружения, то есть число ошибок в зависимости от времени fc (t), полезна по двум причинам — используя программную базу с известными ошибками, можно оценить стратегию, посмотрев, сколько ошибок база позволит обнаружить за данное время. Менеджеры проектов могут добавить fc (t) в модель надежности для оценки того, сколько еще ошибок осталось, и ответить на старый вопрос «Когда заканчивать тестирование?»

## 1.2 Методы тестирования

Тестирование программного обеспечения — это не что иное, как испытание куска кода к контролируемым и неконтролируемым условиям эксплуатации, наблюдение за выходом, а затем изучение, соответствует ли он предварительно определенным условиям. Различные наборы тест-кейсов и стратегий тестирования направлены на достижение одной общей цели - устранение багов и ошибок в коде, и обеспечения точной и оптимальной производительности программного обеспечения.

Широко используемыми методами тестирования являются

* модульное тестирование;
* интеграционное тестирование;
* приемочное тестирование;
* тестирование системы.

Программное обеспечение подвергается этим испытаниям в определенном порядке.

В первую очередь проводится модульный тест. Как подсказывает название, это метод испытания на объектном уровне. Отдельные программные компоненты тестируются на наличие ошибок. Для этого теста требуется точное знание программы и каждого установленного модуля. Таким образом, эта проверка осуществляется программистами, а не тестерами. Для этого создаются тест-коды, которые проверяют, ведет ли программное обеспечение себя так, как задумывалось.

Отдельные модули, которые уже были подвергнуты модульному тестированию, интегрируются друг с другом, и проверяются на наличие неисправностей. Такой тип тестирования в первую очередь выявляет ошибки интерфейса. Интеграционное тестирование можно осуществлять с помощью подхода "сверху вниз", следуя архитектурному сооружению системы. Другим подходом является подход «снизу вверх», который осуществляется из нижней части потока управления.

В системном тестировании, вся система проверяется на наличие ошибок и багов. Этот тест осуществляется путем сопряжения аппаратных и программных компонентов всей системы, и затем выполняется ее проверка. Это тестирование числится под методом тестирования "черного ящика", где проверяются ожидаемые для пользователя условия работы программного обеспечения.

Последним тестом является «Приемочные испытания», который проводится перед передачей программного обеспечения клиенту. Он проводится, чтобы гарантировать, что программное обеспечение, которое было разработано отвечает всем требованиям заказчика. Существует два типа приемо-сдаточных испытаний - то, которое осуществляется членами команды разработчиков, известно, как внутреннее приемочное тестирования (Альфа-тестирование), а другое, которое проводится заказчиком, известно, как внешнее приемочное тестирования. Если тестирование проводится с помощью предполагаемых клиентов, оно называется приемочными испытаниями клиента. В случае если тестирование проводится конечным пользователем программного обеспечения, оно известно, как приемочное тестирование (бета-тестирование).

Также есть несколько основных методов тестирования, которые формируют часть режима тестирования программного обеспечения:

* тестирование методом черного ящика;
* тестирование методом белого ящика;
* тестирование методом серого ящика.

Тестирование методом черного ящика осуществляется без каких-либо знаний внутренней работы системы. Тестер будет стимулировать программное обеспечение для пользовательской среды, предоставляя различные входы и тестируя сгенерированные выходы. Этот тест также известен как Black-box, closed-box тестирование или функциональное тестирование.

Тестирование методом "Белого ящика", в отличие от "черного ящика", учитывает внутреннее функционирование и логику работы кода. Для выполнения этого теста, тестер должен иметь знания кода, чтобы узнать точную часть кода, имеющую ошибки. Этот тест также известен как White-box, Open-Box или Glass box тестирование.

Тестирование методом серого ящика или Gray box тестирование, это что-то среднее между White Box и Black Box тестированием, где тестер обладает лишь общими знаниями данного продукта, необходимыми для выполнения теста. Эта проверка осуществляется посредством документации и схемы информационных потоков. Тестирование проводится конечным пользователем, или пользователям, которые представляются как конечные.

# 2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ УЧЁТ СОТРУДНИКОВ ПРЕДПРИЯТИЯ

## 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 учёта сотрудников предприятия.

Целью тестирования приложения учёта сотрудников предприятия 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 просмотром информации о сотрудниках и приказах.

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 учёта сотрудников предприятия. 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

**Просмотр информации о сотрудниках**

Вывод данных с БД

Добавление/удаление/изменение информации

**Просмотр приказов**

Вывод данных с БД

Добавление/удаление/изменение информации

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

Для C# как правило, проще создать проект модульного теста и заглушки модульных тестов из кода. Кроме того, можно создать проект модульных тестов и тесты вручную в зависимости от потребностей. Если необходимо создать модульный тест из кода на сторонней платформе, то потребуется установить одно из этих расширений: NUnit или xUnit.

Рассмотрим более детально процесс создания проекта модульного теста и заглушек модульных тестов.

В окне редактора кода необходимо выбрать в контекстном меню команду Создать модульные тесты.

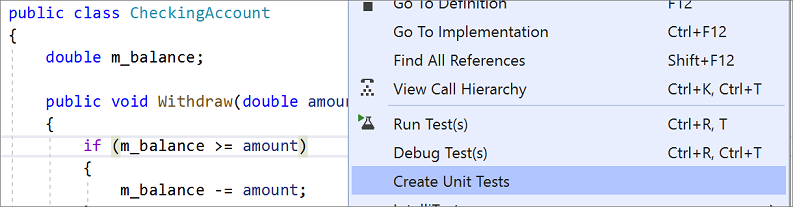


Рисунок 1 – Создание модульного теста

.NET имеет различные разновидности, более формально известные как реализации. .NET 5 + (включая .NET Core) является последней реализацией и работает на любой платформе. платформа .NET Framework является исходной реализацией .net и выполняется только в Windows. Моно используется, когда требуется небольшая среда выполнения. универсальная платформа Windows (UWP) используется для создания современных Windows приложений.

Команда меню Создать модульные тесты доступна только для кода C#. Чтобы использовать этот метод для .NET Core или .NET Standard, требуется Visual Studio 2019 или более поздней версии.

Далее необходимо нажать кнопку ОК, чтобы принять значения по умолчанию для создания модульных тестов, или измените значения, которые использовались для создания и назначения имени проекта модульного теста и модульных тестов. Можно выбрать код, который добавляется по умолчанию в методы модульных тестов.

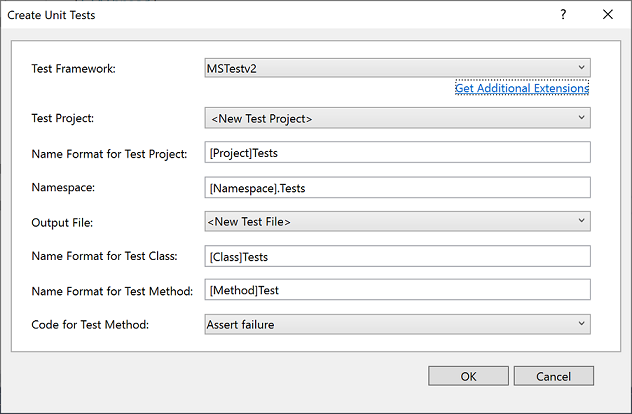


Рисунок 2 – Окно создания модульного теста

Заглушки модульных тестов создаются в новом проекте модульного теста для всех методов в классе.

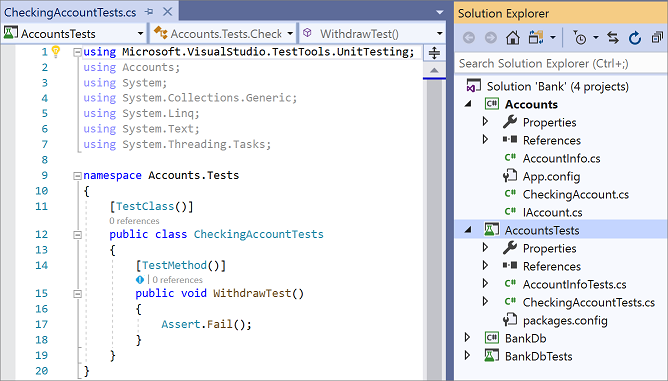


Рисунок 3 – Обозреватель решений с заглушками модульных тестов

Теперь рассмотрим настройку времени ожидания для модульного теста:

Если используется платформа MSTest, можно использовать TimeoutAttribute для установки времени ожидания в отдельном методе теста:

[TestMethod]

[Timeout(2000)] // Milliseconds

public void My\_Test()

{ ...}

Задние лимита времени на максимально разрешенный:

[TestMethod]

[Timeout(TestTimeout.Infinite)] // Milliseconds

public void My\_Test ()

{ ...

}

Рассмотрим выполнение теста в обозревателе тестов и их просмотр.

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).

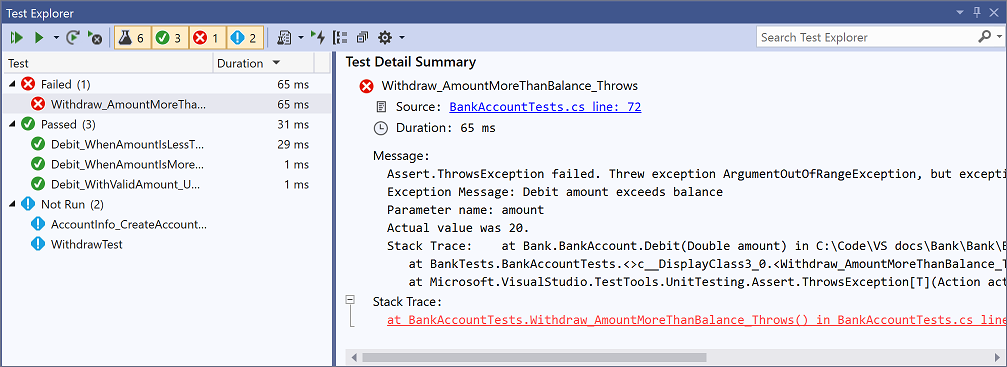


Рисунок 4 – Обозреватель тестов

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.

Кроме того, можно фильтровать тесты по совпадению текста в поле поиска на глобальном уровне или с помощью одного из предустановленных фильтров. Можно запустить любую выборку тестов в любое время. Результаты запущенного теста появляются сразу же в строке "успешно/не успешно" наверху окна обозревателя. Детальная информация результата метода тестирования отображается при выборе теста.

Панель инструментов обозревателя тестов помогает найти, организовать и запустить необходимые тесты.

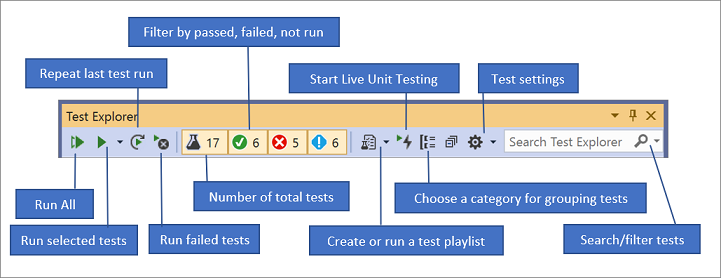


Рисунок 5 – Панель инструментов обозревателя тестов

Можно выбрать Запустить все, чтобы запустить все тесты (или нажать клавиши CTRL + R, V), или выбрать Запустить, чтобы выбрать подмножество тестов для запуска (или нажать клавиши CTRL + R, T). Выберите тест, чтобы просмотреть детальную информацию по нему на панели сведений. Выберите Открыть текст в контекстном меню (клавиша F12) для отображения исходного кода выбранного теста.

Если отдельные тесты не имеют зависимостей, предотвращающих запуск этих тестов в любом порядке, включите параллельное тестирование в меню параметров на панели инструментов. Это может заметно сократить время, необходимое для выполнения всех тестов.

Чтобы запустить модульные тесты после каждой локальной сборки, на панели инструментов обозревателя тестов нужно щелкнуть на значок "Параметры" и выбрать в меню пункт Выполнить тесты после сборки.

Если тестов много, можно отфильтровать список по определенной строке. Для этого введите соответствующий текст в поле поиска обозревателя тестов. Можно ограничить фильтр при помощи выбора фильтров из списка.

Для группировки тестов по категории, нужно нажать кнопку Группировать по.

В случае с ИС «Учёт сотрудников предприятия» был разработан следующий модульный тест:



Рисунок 6 – Модульный тест для ИС «Учёт сотрудников предприятия»

# ЗАКЛЮЧЕНИЕ

В первой части курсовой работы были проанализированы методы тестирования. Тестирование программного обеспечения — это не что иное, как испытание куска кода к контролируемым и неконтролируемым условиям эксплуатации, наблюдение за выходом, а затем изучение, соответствует ли он предварительно определенным условиям. Различные наборы тест-кейсов и стратегий тестирования направлены на достижение одной общей цели - устранение багов и ошибок в коде, и обеспечения точной и оптимальной производительности программного обеспечения.

Были изучены критерии и принципы тестирования информационной системы. Также, были описаны методы тестирования ИС, такие как, метод модульного тестирования, метод интеграционного тестирования, метод приемочного тестирования и метод тестирования системы.

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 сотрудников предприятия».

Практической частью курсовой работы было тестирование фрагмента информационной системы Базы данных.

Подробно описаны и проиллюстрированы функционал и структура самого модуля, а также проведено тестирование готового программного продукта в программе MS Visual Studio 2022.

# 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.