СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc43708868)

[1 Структура предприятия 5](#_Toc43708869)

[2 Оснащенность рабочего места техническими средствами 6](#_Toc43708870)

[3 Анализ использования и функционирования информационных систем на предприятии 8](#_Toc43708871)

[4 Организация резервного копирования на предприятии 9](#_Toc43708872)

[5 Сохранение и восстановление базы данных ИС 10](#_Toc43708873)

[6 Виды документации на программное обеспечение 11](#_Toc43708874)

[7 Виды тестирования информационной системы по объекту тестирования 12](#_Toc43708875)

[8 Проведение тестирования разработанной тестирующей программы 14](#_Toc43708876)

[9 Руководство пользователя 16](#_Toc43708877)

[9.1 Установка системы 16](#_Toc43708878)

[9.2 Использование системы 17](#_Toc43708879)

[ЗАКЛЮЧЕНИЕ 18](#_Toc43708880)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc43708881)

[ПРИЛОЖЕНИЯ 20](#_Toc43708882)

[1. Class program 20](#_Toc43708883)

[2. Class start 21](#_Toc43708884)

[3. Class auth 21](#_Toc43708885)

[4. Class reg 22](#_Toc43708886)

[5. Class welcome 23](#_Toc43708887)

[6. Class stats 24](#_Toc43708888)

[7. Class test 25](#_Toc43708889)

[8. Class result 26](#_Toc43708890)

[9. Class admin 27](#_Toc43708891)

## ВВЕДЕНИЕ

Практика – это один из важнейших элементов профессиональной деятельности. Без неё невозможно закрепить знания и навыки в любой сфере деятельности. Практика позволяет научиться новым профессиональным навыкам и умениям, закрепить уже имеющиеся и проверить их, развить технические и аналитические способности. Производственная практика направлена на усвоение и перенос новых знаний из теории в практику. Практика должна помочь студентам научиться управлять своим временем и распределять задачи по приоритету.

Производственная практика является неотъемлемой частью учебного процесса и направлена на развитие и дальнейшее формирование у студентов практических профессиональных умений и приобретение первоначального практического опыта по профессии ПМ 01 «Эксплуатация и модификация информационных систем». Практика проходит в вычислительном центре ХПЭТ, отделении № 3 с 25 мая 2020 г. до 27 июня 2020 г. и её объём составляет 180 часов.

Основной задачей практики является создание программного продукта и поэтому на ней создавалась система для тестирования «ТеСи», которая является полноценной ИС для тестирования, состоящей из программы и БД.

ТеСи позволяет проводить не какой-то один определённый тест, но любой возможный. Данная система представляет платформу для создания, хранения и прохождения тестов по любым темам любой сложности и размера. Это гораздо эффективнее и удобнее создания множества разрозненных, несвязанных между собой тестов.

Помиго этого практика закрепляет компетенции, приобретаемые обучающимися в результате освоения теоретических курсов, помогает приобрести практический опыт выполнения профессиональных заданий, продолжает формировать общие (универсальные) компетенции обучающихся:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 6. Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.

ОК 7. Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности.

Закреплять профессиональные компетенции:

ПК 1.1. Собирать данные для анализа использования и функционирования информационной системы, участвовать в составлении отчетной документации, принимать участие в разработке проектной документации на модификацию информационной системы.

ПК 1.2. Взаимодействовать со специалистами смежного профиля при разработке методов, средств и технологий применения объектов профессиональной деятельности.

ПК 1.4. Участвовать в экспериментальном тестировании информационной системы на этапе опытной эксплуатации, фиксировать выявленные ошибки кодирования в разрабатываемых модулях информационной системы.

ПК 1.5. Разрабатывать фрагменты документации по эксплуатации информационной системы.

ПК 1.6. Участвовать в оценке качества и экономической эффективности информационной системы.

ПК 1.7. Производить инсталляцию и настройку информационной системы в рамках своей компетенции, документировать результаты работ.

ПК 1.8. Консультировать пользователей информационной системы и разрабатывать фрагменты методики обучения пользователей информационной системы.

ПК 1.9. Выполнять регламенты по обновлению, техническому сопровождению и восстановлению данных информационной системы, работать с технической документацией.

ПК 1.10. Обеспечивать организацию доступа пользователей информационной системы в рамках своей компетенции.

## 1 Структура предприятия

Управление техникумом осуществляется в соответствии с законодательством Российской Федерации и Уставом техникума и осуществляется на основе сочетания принципов единоначалия и коллегиальности.

Непосредственное управление, текущее руководство деятельностью Техникума осуществляет директор, назначаемый Учредителем.

Директор Техникума является единоличным исполнительным органом Техникума и несет ответственность за руководство образовательной, научной, воспитательной работой и организационно-хозяйственной деятельностью Техникума.

Компетенция заместителей директора Техникума устанавливается директором Техникума.

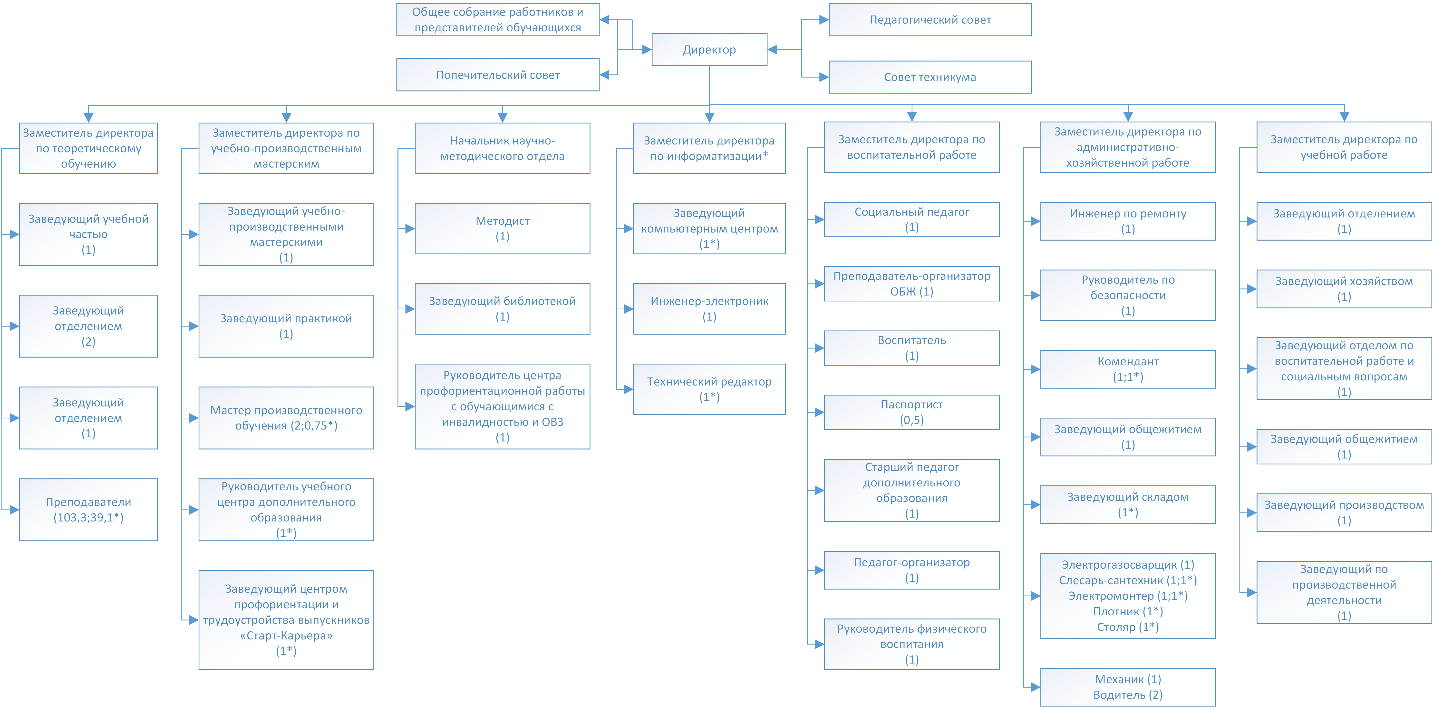


Рисунок 1 – Схема структуры предприятия

## 2 Оснащенность рабочего места техническими средствами

На предприятии было выдано рабочее место в кабинете методиста, за основным комтьютером, в котором можно выделить следующие компоненты:

1. Собственно, сам корпус. Модель: AeroCool P7-C1, белый цвет. Именно в нём и крепятся все компоненты системного блока. Корпусы могут быть очень разных форм, размеров и цветов, но обычно это пластиково-металлический «ящик», в котором предусмотрены крепления для остальных компонентов, а на лицевой стороне есть кнопки для включения / выключения / перезагрузки и некоторые порты (чаще всего USB).
2. Материнская плата. Модель: Gigabyte B450M DS3H. Сложная многослойная печатная и самая большая плата системного блока. К ней подключаются все элементы, и она объединяет все элементы в единую вычислительную систему.
3. Процессор (ЦП / CPU). Модель: AMD Ryzen 5 2600. Вычислительный центр любого ПК. Именно тут и происходит обработка информации. Зачастую именно от ЦП и зависит скорость работы ПК (за исключением случаев, когда другие комплектующие сильно устарели и являются «бутылочным горлышком»).
4. Оперативная память (внутренняя память / ОЗУ). Модель: Kingston HyperX KHX2400C15/8G, 3 плашки, итого: 24 ГБ. Cверхбыстрая память, в которой хранятся обрабатываемые или часто используемые данные. Чем её больше – тем лучше. Также ОЗУ работает в постоянной связке с процессором.
5. Видеокарта. Модель: Gigabyte GeForce 1070 8ГБ. Плата расширения, необходимая для обработки графики. Всё то, что мы видим на экране – результат её работы. Зачастую является самым дорогим компонентом компьютера, т.к. видеокарта – это тоже своего рода компьютер (обладает графическим процессором (ГП), видеопамятью, системой охлаждения, корпусом и основной платой). Также зависит от процессора, т.к. выполняет его команды и поэтому существует понятие «раскрытия» видеокарты, т.е. то, насколько ЦП может задействовать видеокарту.
6. Жёсткий диск (винчестер / HDD) и/или твердотельный накопитель (SSD) – основные, долговременные хранилища информации в компьютере. Именно тут и хранятся ОС, все программы и все данные. Из-за многих особенностей этих устройств (они очень разные), обычно на SSD устанавливают ОС и прикладные программы, а на HDDхранят данные. В данном компьютере был установлен SSD KingStone A400 480ГБ и HDD Western Digital Blue 1ТБ.
7. Блок питания. Модель: AeroCool KCAS-650M. Устройство, необходимое для обеспечения энергией всех остальных компонентов системного блока. Он выделяем им необходимую мощность, не больше, не меньше, например, для процессора критично даже совсем небольшое изменение напряжения. В основном используются блоки питания мощностью от 450 до 650 Вт, однако они бывают и мощнее, до 2-3 КВт.
8. Звуковая карта – плата расширения, необходимая для работы со звуком, например, для его вывода через колонки, или ввода через микрофон. В данном случае она встроена в материнскую плату, и не является необходимостью, хотя и значительно улучшает качетсво звука.
9. Система охлаждения. Собственно, охлаждает все компоненты системного блока. Она может быть, как внешней, так и встроенной (вентилятор в блоке питания). Система охлаждения бывает самых разных видов, но самый распространённая – это активное воздушное охлаждение. Это один или несколько небольших вентиляторов (кулеров), сверху радиатора, которой, в свою очередь прилегает к охлаждаемому компоненту (для лучшего контакта, этот стык заполняется термопастой). Также, зачастую корпус позволяет ставить дополнительные кулеры на себя. Всё это необходимо для поддержания стабильной работы компьютера, т.к. при работе компоненты системного блока сильно нагреваются, и при перегреве отключаются, или даже ломаются.

Помимо собственно системного блока используется периферия:

1. Монитор. Модель: Sumsung S23B350T. Основное устройство отображения зрительной информации. Существуют различные виды и типы мониторов, различающихся по диагонали, герцовке, разрешении и типов матриц.
2. Клавиатура. Модель: Dexp Kite. Устройство ввода информации. Если упростить, то это плата с кнопками, которая необходима для набора текста. Как дополнительные функции могут идти подстветка, эргономичная форма, медиа – клавиши ( например, для упровления звуком ), модульные блоки. Обычно клавиатуры различаются размерами, типами и количеством клавиш.
3. Мышь. Модель: Bloody SP30. Устройство ввода информации. Представляет из себя манипулятор, с помощью которого мы можем указать определённую точку на экране. Используется в комплекте с клавиатурой.
4. Наушники. Модель: Bloody G528C. Устройство вывода звуковой информации. Различаются видами, диапазонами воспроизведения звука, диаметром динамиков и наличием микрофона.

## 3 Анализ использования и функционирования информационных систем на предприятии

На данном предприятии используется две основных информационных системы. Они работают на основе местной локальной сети, которая включает в себя сервер, к которому по топологии звезда подключаются все устройство предприятия. При этом данная локальную сеть является закрытой и допускает вход только изнутри с рабочих компьютеров, причём исключительно с разделением прав.

Первая информационная система – это система резервного копирования. Она сохраняет все важные файлы со всех компьютеров отделения. На каждом устройстве в отделении, есть особые папки, которые предназначены для хранения важных данных и ежедневно в полдень эти данные копируются и отправляются на сервер, где они хранятся в течение месяца.

Вторая информационная система является чисто рабочей и включает в себя несколько других подсистем, которые упрощают работу сотрудникам на предприятии:

1. Рабочий чат с возможностью вложений и отложеных писем. В отличие от проприетарных программ, данная система является полностью закрытой от внешнего доступа и полностью управляемой, что очень сильно повышает общий уровень безопасности.
2. Система полностью электронного документооборота, что позволяет исключить лишние распечатывания документов на бумагу, и наоборот, их сканирование в электронный вариант.
3. Система делегирования и распределения задач на сотрудников с возможностью отслеживания их прогресса выполнения в реальном времени.
4. Система бесшовного wi-fi покрытия по всей территории предприятия, к которой подключаются все беспроводные устройства, что позволяет полностью исключить утечки внутренних данных за пределы локальной сети.
5. Система для руководства, что позволяет отслеживать и оценивать работу, выполняемую сотрудниками, их продуктивность, быстро принимать решения, делегировать задачи любой сложности на сотрудников и ставить сроки их выполнения.

## 4 Организация резервного копирования на предприятии

Резервирование – это процесс создания резервных копий важных данных через определённые промежутки времени для их последующего восстановления в случае повреждений или полной утраты.

Это обычная практика на предприятии не только для исключения потерь хоть сколько-то важных данных, но и для возможности отката на более раннюю версию.

На очень крупных IT-предприятиях используются свои системы резервного копирования и восстановления, которые включают в себя высокопропускную локальную сеть, сервера для хранения копий и программные системы для создания и категоризации копий.

Важнейшие параметры резервного копирования – это частота, время сохранения и список сохраняемых файлов.

Частота определяет максимальный срок потери данных. Например, если резервация будет проходить раз в день, то максимальный срок отката данных – день. Но при этом, если откатить данные через 2 часа после резервного копирования, то и срок отката составит 2 часа. Чем важнее и критичнее данные, тем чаще они должны сохранятся, особенно если они часто обновляются.

Время сохранения определяет максимальный срок отката данных. Если архивы сохраняются на месяц, то дальше этого срока откатиться будет невозможно. Чем важнее данные, тем на дольший срок их надо сохраняться.

Список сохраняемых файлов определяет, что конкретно сохраняется. Сохранять абсолютно всё нелогично и даже невыгодно, а то и выгодно. Поэтому сохраняются самые важные данные, в первую очередь те, от которых зависит работоспособность системы.

## 5 Сохранение и восстановление базы данных ИС

Сохранение и восстановление БД – это по-сути, резервное копирование и восстановление БД. Обычно оно представляет преобразование всей БД или отдельной её части в последовательность SQL-запросов с помощью выполнения которых её можно в точности воссоздать – SQL-дамп.

Это является полным восстановлением всей базы данных, либо её отдельных частей.

Помимо этого, в современных СУБД используются также другие подсистемы и хитрости для обеспечения сохранности данных.

Современные базы данных также часто выполняют различные транзакции, то есть непрерывные последовательности sql-запросов, объединенные в одну цепочку. И в случае нарушения хотя бы одного из элементов этой цепочке, вся транзакция полностью отменяется и все данные полностью восстанавливаться обратно.

Также большинство современных СУБД поддерживает объединение в кластер и репликацию, то есть, проще говоря, дублирование одной базы данных на другую или несколько других. И в даже случае физического отключения одного из элементов такого кластера, управление случайным образом перейдет к одной из работающих на данный момент баз данных и работа кластера в целом продолжится. При этом данные не потеряются.

Помимо этого, такая архитектура позволяет создавать огромные базы данных, рассчитаные на колоссальные объемы. Однако такой подход имеет и некоторые проблемы, в частности, требуется наличие сверхскоростной магистрали для связи между узлами кластера, а также мощную синхронизацию запросов в кластере для исключения ошибок синхронизации данных.

Поскольку в большинстве информационных систем база данных является главным, а порой даже единственным хранилищем данных, вопросами её резервного копирования занимаются в первую очередь.

При этом сохранение базы данных может происходить как в автоматическом режиме, так и в ручном. Также поддерживаются сохранения различных уровней. Можно сохранить всю базу данных, можно сохранить отдельно таблицу, можно сохранить только данные из таблицы, можно сохранить только структуру таблицы и так далее.

## 6 Виды документации на программное обеспечение

Существует четыре основных типа документации на ПО:

1. Архитектурная/проектная — обзор программного обеспечения, включающий описание рабочей среды и принципов, которые должны быть использованы при создании ПО.

Описание продукта в общих чертах. Условно – обоснованный ответ на вопрос «почему именно так, а не иначе». Описание скелета проекта, основных схем, решений и выборов.

1. Техническая — документация на код, алгоритмы, интерфейсы, API.

Описание внутреней и внешней логики кода, подробное описание алгоритмов, блок-схем. Условно – ответ на вопрос «как это устроено». Очень точное описание всех производимых действий системой, вплоть до каждой, сколь-либо существенной мелочи.

1. Пользовательская — руководства для конечных пользователей, администраторов системы и другого персонала.

Описания интерфейсов, внешних функций, возможностей программы для конечного пользователя. Условно – ответ на вопрос «что оно может и как». Простые пошаговые описания сценариев действий с программой, предусмотренные ответы на частые возможные ответы, встроенные меню помощи и база знаний. В некоторых случаях – встроенная система обучения.

1. Маркетинговая – рекламные материалы.

Рекламные материалы, направленные на завлечение пользователей и / или оплату дополнительных услуг с этим связанных. Условно – ответ на вопрос «зачем оно нужно и что даст». В это входит полная «упаковка» продукта от названия и описания до внешнего вида и слогана.

## 7 Виды тестирования информационной системы по объекту тестирования

Испытание информационной системы и тестирование программного продукта на первый взгляд одно и то же, но на практике это не совсем так. Если учесть, что информационная система – это не только используемые в ее составе программные компоненты, но и аппаратное и организационное обес-печение, то и в результатах ее испытаний должны быть отражены показатели выбранных серверов, рабочих станций, сетевого оборудования (их надеж-ность и производительность), а также эффективность разработанного регла-мента эксплуатации системы.

Выделяют следующие виды и подвиды тестирования ИС:

1. Функциональное тестирование ( сравнение количества и качества ожидаемых и предоставляемых функций ):
   1. Функциональное тестирование – проверка соответствия системы заявленным функциональным требованиям.
   2. Регрессионное функциональное тестирование – проверка работоспособности системы, уже подвергавшейся процедуре функционального тестирования, после внесения в систему или ее инфраструктуру каких-либо изменений.
   3. Интеграционное тестирование – испытания, направленные на выявление проблем взаимодействия отдельных компонентов системы.
   4. Тестирование пользовательских интерфейсов (юзабилити) с помощью целевой группы – проверка удобства и простоты использования системы с помощью фокус-группы.
   5. Экспертное тестирование пользовательских интерфейсов (юзабилити) – экспертная проверка удобства и простоты использования системы.
   6. Приемочное тестирование – комплексное испытание информационной системы, выполняемое представителями заказчика по специально разработанной исполнителем программе и методике испытаний.
2. Нефункциональное тестирование ( сравнение предоставляемых производительности, удобности, функциональности, эргономичности по сравнению с ожидаемым ):
   1. Нагрузочное тестирование – испытание информационной системы в условиях прогнозируемой нормальной нагрузки. Проще говоря тестирование на отказоустойчивость под нагрузкой на протяжение длительного времени.
   2. Тестирование надежности и отказоустойчивости – испытание информационной системы в условиях минимальных аппаратных ресурсах и максимально допустимой нагрузки. Проверка работоспособность системы в стрессовых ситуациях.
   3. Синтетическое тестирование – стандартизированная проверка производительности серверного оборудования системы.
3. Автоматизированное тестирование:
   1. Автоматизированное регрессионное тестирование – проверка работоспособности системы после внесения в нее или ее инфраструктуру каких-либо изменений.
   2. Автоматизированное смоук-тестирование на промышленной среде – проверка работоспособности ключевых функций системы после внесения в нее или ее инфраструктуру каких-либо изменений.
4. Инсталляционное тестирование:
   1. Инсталляционное тестирование – проверка возможности развертывания и конфигурирования системы из дистрибутива.
   2. Тестирование методического обеспечения – проверка актуальности методического обеспечения системы и его соответствия всем заявленным требованиям.

## 8 Проведение тестирования разработанной тестирующей программы

Тестирование ПО — проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. Тестирование включает следующие этапы:

1. Планирование работ. Определение входов и выходов программы, создание тестов на основе логики работы или функционала.
2. Проектирование тестов путем ручной разработки или автоматической генерации. Тесты создаются для их автоматического проведения. При этом для тестирования различных видов программ используются разные виды тестирование.
3. Выполнение тестирования с получением результатов. Собственно тестирование программы в разных тестах и условиях и сбор статистики.
4. Анализ полученных результатов выполнения с целью оценки качества ПО и получения данных о багах, ошибках, их критичности и необходимости исправления.

Это полноценное тестирование, используемое на предприятиях. Однако для более мелких проектов или напротив, огромных сетевых проектов могут применятся другие, более подходящие типы тестирования.

В случае с ТеСи тестирование производилось следующими путями:

1. Тестирование и исправление средствами IDE. Данный вид тестирования происходит постоянно во время написания кода и поправляет самые критические ошибки. Иначе программа просто не заработает. Но в зависимости от IDE или редактора могут правиться и выявляться ошибки и более низких уровней – вплоть до поставленной не в том месте запятой или точки. Вручную же их очень сложно заметить.
2. Тестирование прямо во время создания. Проверка отдельных элементов и кусков кода с их мгновенными правками и исправлениями. В большинстве своём это исправляет большую часть ошибок, например, таких как синтаксические и логические.
3. Отладка с помощью сообщений от ошибок и предупреждений от собственно платформы разработки. Это не даст идельного кода, но позволит не допускать грубых и недопустимых ошибкок.
4. Общее тестирование после прохождение какого-либо этапа разработки. Например, общее тестирование после создания новой формы. Это избавляет от большой части ошибок связанных с взаимодействиями элементов, проблем с интерфейсом и связями форм.
5. Отладка с помощью логов. Это позволяет отслеживать логику программы и выполняемую, пусть и не всегда логичную последовательность действий, а значит и исправлять ошибки с этим связанные.
6. Версионное тестирование. Альфа-тестирование ( силами разработчиков ), бета-тестирование ( силами бета-тестеров ), релизное тестирование ( силами пользователей ). Позволяет выявлять самые неочивидные и неожиданные баги. Выигрывает не благодаря качеству тестирования, но количеству и нелогичности. Реальный пользователь не всегда поступает так, как задумано и это позволяет находить самые неочевидные ошибки и несостыковки. Поэтому чем дольше программой пользуются, тем меньше в ней остаётся багов.

Также тестирование – задокументированный процесс, поэтому существуют стандарты, относящиеся к тестированию и по которым оно производится:

1. IEEE 829—2008 IEEE Standard for Software and System Test Documentation
2. ANSI/IEEE Std 1008—1987 — IEEE Standard for Software Unit Testing
3. ISO/IEC/IEEE 29119-1:2013 Software and systems engineering — Software testing — Part 1: Concepts and definitions
4. ISO/IEC/IEEE 29119-2:2013 Software and systems engineering — Software testing — Part 2: Test processes
5. ISO/IEC/IEEE 29119-3:2013 Software and systems engineering — Software testing — Part 3: Test documentation

## 9 Руководство пользователя

## Установка системы

Для того, чтобы развернуть тестирующую систему с нуля, требуется:

1. Развернуть MySQL версии 8.0 или выше на сервере или локально. Размещение на сервере позволяет централизовано подключать большое количество клиентов к системе и управлять всем из одного места. Размещение локально, напротив, даёт возможность изолированной работы, быстрого развёртывания и изменения. Параметры для подключения в программе по умолчанию:
   1. ip: localhost
   2. порт: 3306
   3. пользователь: root
   4. пароль: root
   5. название базы данных: practic\_test
2. Развернуть базу данный из sql-дампа. Дамп присутствует в папке sql\_dump в двух возможных вариантах:
   1. с данными ( заранее включёнными тестами ) - full\_dump.sql
   2. без данных ( только структура ) - only\_structure\_dump.sql
3. Пересобрать программу и изменить исходники ( по необходимости ). Например, для того чтобы изменить параментры подключения к СУБД. Однако если используется локальная версия MySQL 8.0 с параметрами по умолчанию, этот шаг можно пропустить. Техническая информация по проекту:
   1. используемый язык: C#
   2. используемая платформа: .NET Framework 4.8+
   3. используемая технология: Windows Forms
   4. рекомендуемый редактор: Visual Studio
   5. расположение исходников: /src/Test/
   6. параметы подключнения к БД: /src/Test/dbMySQL.cs
4. Установить программу клиентам ( пользователям ) и обеспечить доступ к БД. Доступены как установщик, так и портативная версия. Требования для работы:
   1. доступная связь с БД
   2. установленный .NET Framework 4.8+
   3. ОС Windows 7 и выше

## 9.2 Использование системы

После того, как всё развёрнуто и установлено, доступно собственно использование ИС. Система подразумевает три сценария развития событий:

1. Собственно использование, т.е. прохождение тестов в системе. Для этого нужно лишь:
   1. Войти в систему ( предварительно зарегистрировавшись )
   2. Выбрать тест
   3. Пройти тест
   4. Получить результат
   5. При необходимости – изучить статистику и сравнить результаты
2. Добавление тестов и их редактирование. Для этого необходимо обладать учётной записью с правами администратора. Порядок действий при этом следующий:
   1. Вход в систему с правами админа
   2. Переход в редактор
   3. Выбор готового / создание нового теста
   4. Изменение / добавление вопросов в тесте
   5. Сохранение готового теста в системе
3. Отладка / редактирование собственно системы. Для этого предусмотренно следующие:
   1. Наличие дампа с исключительно структурой БД
   2. Комментарии в коде и разнесённая структура
   3. Мощная система логов с различными уровнями

## ЗАКЛЮЧЕНИЕ

Практика проходила в вычислительном центре ХПЭТ, отделении № 3 с 25 мая 2020 г. до 27 июня 2020 г. и её объём составил 180 часов. Центр был оснащён всеми необходимыми средствами вычислительной техники и учебного оборудования, необходимых для прохождения практики.

По прохождению учебной практики были успешно выполнены все цели и задачи по закреплению теоретических и формированию практических знаний, а также по внедрению профессионально необходимых качеств по профессии «Эксплуатация и модификация информационных систем».

Основной целью было создание тестирующий системы «ТеСи», всех её модулей и компонентов, для дальнейшего использования во всех сферах деятельности, где может требоваться проверка знаний. В итоге тестирующая система была создана в полном объёме со всеми необходимыми для её работы модулями и протестирована на других студентах из группы практикантов.

Второй основной целью было обучение работе на предприятии, изучение специального программного и технологического обеспечения, получение опыта с информационными системами, применяемыми при работе с реальными задачами.

Помимо этого, во время практики были закреплены следующие теоретические знания и практические навыки:

1. Должностные инструкции и требования, т.е. необходимые для соблюдения правила на время практики. Должностные инструкции были освоены и полностью соблюдены.
2. Составление целей и задач, которые нужно выполнить для того, чтобы успешно создать тестирующую систему. Весь список поставленных целей и задач был успешно выполнен.
3. Составление плана-графика на время практики. Выполнение всех задач произошло точно по плану и в срок.
4. Профессиональная работа с СУБД MySQL и её связь с ЯП C# с использованием платформы .NET Framework и технологии Windows Forms в IDE Microsoft VS.
5. Ознакомление с работой на предприятии. Было проведено полное освоение всех выполняемых рабочих задач.
6. Изучение системного, программного и технического обеспечения, используемого на предприятии при решение реальных задач.

В итоге прохождения практики все поставленные цели и задачи были выполнены, теоретические знания получены и освоены, а практические навыки закреплены.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 2.104 ЕСКД для всех специальностей.
2. ГОСТ 7.1-2003 Система стандартов по информации.
3. ГОСТ 19.781-90 «Единая система программной документации. Программное обеспечение систем обработки информации. Термины и определения».
4. ГОСТ 34.321-96 Информационные технологии.
5. ГОСТ 34.601-90 Автоматизированные системы. Стадии создания
6. ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы
7. Стиллмен Э., Грин Дж. Изучаем C#. 3-е издание.: Издательство Питер СПб.
8. Албахари Дж., Албахари Б. C# 7.0. Карманный справочник. Скорая помощь для программистов на C# 7.0.: Издательство Альфа-Книга
9. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд.: Издательство Питер
10. Бэрон Шварц, Петр Зайцев, Вадим Ткаченко, Джереми Заводны, Арьен Ленц, Дерек Боллинг MySQL. Оптимизация производительности.: Издательство Символ-Плюс
11. Документация по .NET [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/
12. Структура и функции операционной системы [Электронный ресурс]. Режим доступа: https://studfiles.net/preview/5946525/page:11/

# ПРИЛОЖЕНИЯ

Данный листинг приводит только классы с бизнес-логикой и не затрагивает классы с отрисовкой графических элементов форм, а также классы внешних модулей и библиотек, в частности для логгирования и работы с базой данных. Также он не включает дополнительные файлы, такие как ресурсы, конфигурации и динамически загружаемые библиотеки. Классы приведены полностью, без сокращений и вырезанных блоков кода.

## **Class program**

using NLog;

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace Test {

static class Program {

[STAThread]

static void Main() {

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new start());

}

public static void ini() {

data.numQuest=0;

data.numTrueAnswer=0;

data.isCheck=false;

data.score=0;

data.timeSec=0;

data.timeMin=0;

data.timeHour=0;

data.quests.Clear();

data.ids.Clear();

}

public static class data {

public static string idTest { get; set; } // id теста

public static string test { get; set; } // название теста

public static string name { get; set; } // имя пользователя

public static bool isCheck { get; set; } // выбран ли ответ?

public static int numQuest { get; set; } // текущий номер вопроса

public static int numTrueAnswer { get; set; } // правильный ответ на текущий вопрос

public static int score { get; set; } // счёт ответов

public static int timeSec { get; set; } // секунды

public static int timeMin { get; set; } // минуты

public static int timeHour { get; set; } // часы

public static string idUser { get; set; } // id пользователя

public static string login { get; set; } // ник

public static string passHash { get; set; } // хэш пароля

public static string status { get; set; } // статус пользователя

public static bool isAuth { get; set; } // аутентифицирован ли пользователь?

public static List<string> quests=new List<string>(); // список вопросов

public static List<int> ids=new List<int>(); // список id вопросов

}

}

}

## **Class start**

using NLog;

using System;

using System.Windows.Forms;

namespace Test {

public partial class start : Form {

public start() {

InitializeComponent();

}

private void button\_primary\_Click(object sender,EventArgs e) {

start.ActiveForm.Hide();

auth auth=new auth();

auth.ShowDialog();

Close();

}

private void button1\_Click(object sender,EventArgs e) {

start.ActiveForm.Hide();

welcome welcome=new welcome();

welcome.ShowDialog();

Close();

}

private void button\_danger\_Click(object sender,EventArgs e) {

Close();

}

}

}

## **Class auth**

using MySql.Data.MySqlClient;

using NLog;

using System;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

namespace Test {

public partial class auth : Form {

public auth() {

InitializeComponent();

string currentPath=AppDomain.CurrentDomain.BaseDirectory;

if (!Directory.Exists(Path.Combine(currentPath,"userData"))) {

Directory.CreateDirectory(Path.Combine(currentPath,"userData"));

}

try {

string[] arrStr= File.ReadAllLines(currentPath+"/userData/data.txt");

loginField.Text=cryptoPass.EasyDecryptPassword(arrStr[0]);

passField.Text=cryptoPass.EasyDecryptPassword(arrStr[1]);

checkSavePass.Checked=true;

}

catch {

File.Create(currentPath+"/userData/data.txt");

}

}

private void loginButton\_Click(object sender,EventArgs e) {

string login=loginField.Text;

string pass=passField.Text;

dbMySQL db=new dbMySQL();

MySqlCommand command=new MySqlCommand("SELECT `pass\_hash`,`name`,`id`,`status` FROM `users` WHERE `login`=@login",db.getConnection());

command.Parameters.Add("@login",MySqlDbType.VarChar).Value=login;

db.openConnection();

try {

using (MySqlDataReader readerCommand=command.ExecuteReader()) {

if (readerCommand.Read()) {

if (BCrypt.Net.BCrypt.Verify(pass,readerCommand[0].ToString())) {

if (checkSavePass.Checked == true) {

string currentPath=AppDomain.CurrentDomain.BaseDirectory;

FileStream fs=File.Create(currentPath+"/userData/data.txt");

StreamWriter writer=new StreamWriter(fs);

string localLoginHash=cryptoPass.EasyEncryptPassword(login);

string localPassHash=cryptoPass.EasyEncryptPassword(pass);

writer.Write(localLoginHash+"\n"+localPassHash);

writer.Close();

}

string passHash=BCrypt.Net.BCrypt.HashPassword(pass);

string name =readerCommand[1].ToString();

string id =readerCommand[2].ToString();

string status =readerCommand[3].ToString();

Program.data.idUser=id;

Program.data.login=login;

Program.data.passHash=passHash;

Program.data.name=name;

Program.data.status=status;

Program.data.isAuth=true;

auth.ActiveForm.Hide();

welcome welcome=new welcome();

welcome.ShowDialog();

Close();

}

else {

return;

}

}

else {

return;

}

}

}

catch {

Program.err(res.err\_dbNotQuery);

}

db.closeConnection();

}

private void text\_alert\_secondary\_Click(object sender,EventArgs e) {

auth.ActiveForm.Hide();

reg reg=new reg();

reg.ShowDialog();

Close();

}

}

}

## **Class reg**

using MySql.Data.MySqlClient;

using NLog;

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Test {

public partial class reg : Form {

public reg() {

InitializeComponent();

}

private void loginButton\_Click(object sender,EventArgs e) {

string name=nameField.Text;

string login=loginField.Text;

string email=emailField.Text;

string pass=passField.Text;

string pass2=pass2Field.Text;

string passHash=BCrypt.Net.BCrypt.HashPassword(pass);

dbMySQL db=new dbMySQL();

db.openConnection();

MySqlCommand checkUser=new MySqlCommand("SELECT `login` FROM `users` WHERE login=@login OR email=@email",db.getConnection());

checkUser.Parameters.Add("@login",MySqlDbType.VarChar).Value=login;

checkUser.Parameters.Add("@email",MySqlDbType.VarChar).Value=email;

try {

using (MySqlDataReader reader=checkUser.ExecuteReader()) {

if (reader.Read()) {

return;

}

else {

reader.Close();

MySqlCommand insertUser=new MySqlCommand("INSERT INTO `users` (`login`,`name`,`pass\_hash`,`email`) VALUES (@login,@name,@passHash,@email)",db.getConnection());

insertUser.Parameters.Add("@login",MySqlDbType.VarChar).Value=login;

insertUser.Parameters.Add("@name",MySqlDbType.VarChar).Value=name;

insertUser.Parameters.Add("@passHash",MySqlDbType.VarChar).Value=passHash;

insertUser.Parameters.Add("@email",MySqlDbType.VarChar).Value=email;

}

}

}

catch {

Program.err(res.err\_dbNotQuery);

}

db.closeConnection();

}

private void tp\_auth\_form\_Click(object sender,EventArgs e) {

reg.ActiveForm.Hide();

auth auth=new auth();

auth.ShowDialog();

Close();

}

}

}

## **5.** **Class welcome**

using MySql.Data.MySqlClient;

using NLog;

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Test {

public partial class welcome : Form {

public welcome() {

InitializeComponent();

dbMySQL db=new dbMySQL();

Program.ini();

MySqlCommand command=new MySqlCommand("SELECT id,name,description FROM tests LIMIT 10",db.getConnection());

db.openConnection();

logger.Debug(" "+res.db\_executeQuery);

try {

using (MySqlDataReader readerCommand=command.ExecuteReader()) {

int i=0;

while (readerCommand.Read()) {

Label H2=new Label();

Button button\_start=new Button();

Label H4=new Label();

Panel panel=new Panel();

Controls.Add(panel);

i++;

}

}

}

catch {

Program.err(res.err\_dbNotQuery);

}

db.closeConnection();

if (Program.data.isAuth == true) {

authNameLabel.Visible=true;

authNameField.Visible=true;

authNameField.Text=Program.data.name;

}

else {

nameLabel.Visible=true;

nameField.Visible=true;

}

if (Program.data.status == "a") {

adminButton.Visible=true;

}

}

private void button\_start\_Click(object sender,EventArgs e) {

if (Program.data.isAuth == false) {

if (nameField.Text == "") {

nameField.BackColor=Color.Red;

return;

}

else {

Program.data.name=nameField.Text;

}

}

Program.data.idTest=(sender as Button).Parent.Name;

Program.data.test=((sender as Button).Parent.Controls[2] as Label).Text;

welcome.ActiveForm.Hide();

test Test=new test();

Test.ShowDialog();

Close();

}

private void button\_info\_Click(object sender,EventArgs e) {

welcome.ActiveForm.Hide();

stats stats=new stats();

stats.ShowDialog();

Close();

}

private void button1\_Click(object sender,EventArgs e) {

welcome.ActiveForm.Hide();

admin admin=new admin();

admin.ShowDialog();

Close();

}

}

}

## **Class stats**

using MySql.Data.MySqlClient;

using NLog;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace Test {

public partial class stats : Form {

public stats() {

InitializeComponent();

dbMySQL db=new dbMySQL();

MySqlCommand testsCom=new MySqlCommand("SELECT `id\_test`,COUNT(\*) FROM `logs` GROUP BY `id\_test` HAVING COUNT(\*) >= 5",db.getConnection());

db.openConnection();

List<int> tests=new List<int>();

using (MySqlDataReader readerTestsCom=testsCom.ExecuteReader()) {

while (readerTestsCom.Read()) {

tests.Add(int.Parse(readerTestsCom[0].ToString()));

}

}

for (int i=0; i < tests.Count; i++) {

Label label0=new Label();

Label label1=new Label();

Label label2=new Label();

Label label3=new Label();

Label label4=new Label();

Label label5=new Label();

Panel panel=new Panel();

panel.Controls.Add(label0);

panel.Controls.Add(label1);

panel.Controls.Add(label2);

panel.Controls.Add(label3);

panel.Controls.Add(label4);

panel.Controls.Add(label5);

MySqlCommand stats=new MySqlCommand("SELECT `id\_test`,`name`,`username`,`result`,`mark` FROM `logs`,`tests` WHERE `id\_test`=`tests`.`id` AND `id\_test`=@id ORDER BY `result` DESC LIMIT 5 ",db.getConnection());

stats.Parameters.Add("@id",MySqlDbType.Int32).Value=tests[i];

try {

using (MySqlDataReader readerStats=stats.ExecuteReader()) {

int j=1;

while (readerStats.Read()) {

(panel.Controls[j] as Label).Text=readerStats[2].ToString()+" - "+readerStats[3].ToString()+" баллов ("+readerStats[4].ToString()+")";

logger.Debug(" - "+readerStats[2].ToString()+" - "+readerStats[3].ToString()+" баллов ("+readerStats[4].ToString()+")");

j++;

}

}

}

catch {

Program.err(res.err\_dbNotQuery);

}

Controls.Add(panel);

}

db.closeConnection();

}

private void button\_info\_Click(object sender,EventArgs e) {

stats.ActiveForm.Hide();

welcome welcome=new welcome();

welcome.ShowDialog();

Close();

}

}

}

## **Class test**

using MySql.Data.MySqlClient;

using NLog;

using System;

using System.Linq;

using System.Windows.Forms;

namespace Test {

public partial class test : Form {

readonly dbMySQL db=new dbMySQL();

public test() {

InitializeComponent();

H1.Text=Program.data.test;

timer.Start();

Program.ini();

MySqlCommand quests=new MySqlCommand("SELECT id,question FROM questions WHERE id\_test=@id",db.getConnection());

quests.Parameters.Add("@id",MySqlDbType.Int32).Value=Program.data.idTest;

db.openConnection();

logger.Debug(res.db\_executeQueryFirst);

try {

using (MySqlDataReader readerQuests=quests.ExecuteReader()) {

while (readerQuests.Read()) {

Program.data.ids.Add(int.Parse(readerQuests[0].ToString()));

Program.data.quests.Add(readerQuests[1].ToString());

}

Program.data.numQuest=0;

}

}

catch {

Program.err(res.err\_dbNotQuery);

}

Random random=new Random();

for (int i=Program.data.ids.Count - 1; i >= 1; i--) {

int j=random.Next(i+1);

var temp=Program.data.ids[j];

Program.data.ids[j]=Program.data.ids[i];

Program.data.ids[i]=temp;

var temp2=Program.data.quests[j];

Program.data.quests[j]=Program.data.quests[i];

Program.data.quests[i]=temp2;

}

new\_quest();

}

private void button\_success\_Click(object sender,EventArgs e) {

if (Program.data.isCheck) {

if (Program.data.quests.Count() >= Program.data.numQuest) {

if ((radioGroup.Controls[Program.data.numTrueAnswer] as RadioButton).Checked == true) {

Program.data.score++;

}

rb\_unvisible();

new\_quest();

}

}

}

private void rb0\_CheckedChanged(object sender,EventArgs e) {

RadioButton radioButton=(RadioButton)sender;

if (radioButton.Checked) {

Program.data.isCheck=true;

}

else {

Program.data.isCheck=false;

}

}

public void new\_quest() {

if (Program.data.quests.Count() > Program.data.numQuest) {

MySqlCommand answers=new MySqlCommand("SELECT variant,answer FROM variants WHERE id\_question=@id\_quest",db.getConnection());

answers.Parameters.Add("@id\_quest",MySqlDbType.Int32).Value=Program.data.ids[Program.data.numQuest];

H2.Text=Program.data.quests[Program.data.numQuest];

try {

using (MySqlDataReader readerAnswers=answers.ExecuteReader()) {

int i=0;

while (readerAnswers.Read()) {

(radioGroup.Controls[i] as RadioButton).Text=readerAnswers[0].ToString();

(radioGroup.Controls[i] as RadioButton).Visible=true;

if (readerAnswers[1].ToString() == "t") {

Program.data.numTrueAnswer=i;

}

i++;

}

}

}

catch {

Program.err(res.err\_dbNotQuery);

}

Program.data.numQuest++;

}

else {

timer.Stop();

test.ActiveForm.Hide();

result result=new result();

result.ShowDialog();

Close();

}

}

private void button\_danger\_Click(object sender,EventArgs e) {

timer.Stop();

Close();

}

private void timer\_Tick(object sender,EventArgs e) {

Program.data.timeSec++;

if (Program.data.timeSec > 59) {

Program.data.timeSec=0;

Program.data.timeMin++;

}

if (Program.data.timeMin > 59) {

Program.data.timeMin=0;

Program.data.timeHour++;

}

timerSec.Text=Program.data.timeSec.ToString();

timerMin.Text=Program.data.timeMin.ToString();

timerHour.Text=Program.data.timeHour.ToString();

}

}

}

## **Class result**

using MySql.Data.MySqlClient;

using NLog;

using System;

using System.Linq;

using System.Windows.Forms;

namespace Test {

public partial class result : Form {

public result() {

InitializeComponent();

H1.Text=Program.data.test;

dbMySQL db=new dbMySQL();

int pers=(int)((float)Program.data.score / Program.data.quests.Count() \* 100);

resultLabel.Text=Program.data.score+" / "+Program.data.quests.Count()+" ("+pers+"%)";

TimeSpan time=new TimeSpan(0,Program.data.timeMin,Program.data.timeSec);

int mark;

if (pers > 90) {

markLabel.Text=res.mark\_5;

}

else if ((pers > 75) && (pers <= 90)) {

markLabel.Text=res.mark\_4;

}

else if ((pers > 50) && (pers <= 75)) {

markLabel.Text=res.mark\_3;

}

else {

markLabel.Text=res.mark\_2;

}

string sql;

if (Program.data.isAuth == true) {

sql="INSERT INTO `practic\_test`.`logs` (`id\_test`,`id\_user`,`result`,`mark`,`time`) VALUES (@id,@iduser,@score,@mark,@time);";

}

else {

sql="INSERT INTO `practic\_test`.`logs` (`id\_test`,`username`,`result`,`mark`,`time`) VALUES (@id,@name,@score,@mark,@time);";

}

MySqlCommand addLog=new MySqlCommand(sql,db.getConnection());

addLog.Parameters.Add("@id",MySqlDbType.Int32).Value=Program.data.idTest;

addLog.Parameters.Add("@score",MySqlDbType.Int32).Value=Program.data.score;

addLog.Parameters.Add("@mark",MySqlDbType.Int32).Value=mark;

addLog.Parameters.Add("@time",MySqlDbType.Time).Value=time;

if (Program.data.isAuth == true) {

addLog.Parameters.Add("@iduser",MySqlDbType.Int32).Value=Program.data.idUser;

}

else {

addLog.Parameters.Add("@name",MySqlDbType.VarChar).Value=Program.data.name;

}

db.openConnection();

try {

addLog.ExecuteNonQuery();

}

catch {

Program.err(res.err\_dbNotQuery);

}

db.closeConnection();

}

private void button\_success\_Click(object sender,EventArgs e) {

result.ActiveForm.Hide();

welcome welcome=new welcome();

welcome.ShowDialog();

Close();

}

private void button\_danger\_Click(object sender,EventArgs e) {

Close();

}

private void button\_warning\_Click(object sender,EventArgs e) {

result.ActiveForm.Hide();

test test=new test();

test.ShowDialog();

Close();

}

}

}

## **9.** **Class admin**

using MySql.Data.MySqlClient;

using NLog;

using System;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace Test {

public partial class admin : Form {

readonly dbMySQL db=new dbMySQL();

public admin() {

InitializeComponent();

Program.ini();

}

private void admin\_Load(object sender,EventArgs e) {

this.testsTableAdapter.Fill(this.practic\_testDataSet.tests);

}

private void comboBox1\_SelectedIndexChanged(object sender,EventArgs e) {

panelMain.Controls.Clear();

Program.ini();

db.openConnection();

logger.Debug(res.db\_createQuery);

MySqlCommand quests=new MySqlCommand("SELECT id,question FROM questions WHERE id\_test=@idtest",db.getConnection());

quests.Parameters.Add("@idtest",MySqlDbType.VarChar).Value=comboBox1.SelectedValue;

try {

using (MySqlDataReader readerQuests=quests.ExecuteReader()) {

while (readerQuests.Read()) {

Program.data.ids.Add(int.Parse(readerQuests[0].ToString()));

Program.data.quests.Add(readerQuests[1].ToString());

}

Program.data.numQuest=0;

}

}

catch {

Program.err(res.err\_dbNotQuery);

}

while (Program.data.quests.Count() > Program.data.numQuest) {

MySqlCommand answers=new MySqlCommand("SELECT variant,answer FROM variants WHERE id\_question=@id\_quest",db.getConnection());

answers.Parameters.Add("@id\_quest",MySqlDbType.Int32).Value=Program.data.ids[Program.data.numQuest];

TextBox answersLabel=new TextBox();

TextBox questLabel =new TextBox();

Panel panel1 =new Panel();

RadioButton radioButton1=new RadioButton();

RadioButton radioButton2=new RadioButton();

RadioButton radioButton3=new RadioButton();

RadioButton radioButton4=new RadioButton();

RadioButton radioButton5=new RadioButton();

RadioButton radioButton6=new RadioButton();

RadioButton radioButton7=new RadioButton();

questLabel.Text=Program.data.quests[Program.data.numQuest];

try {

using (MySqlDataReader readerAnswers=answers.ExecuteReader()) {

int i=0;

while (readerAnswers.Read()) {

answersLabel.Text += readerAnswers[0].ToString()+Environment.NewLine;

if (readerAnswers[1].ToString() == "t") {

(panel1.Controls[i] as RadioButton).Checked=true;

}

i++;

}

answersLabel.Text=answersLabel.Text.Remove(answersLabel.Text.Length - 2);

}

}

catch {

Program.err(res.err\_dbNotQuery);

}

panelMain.Controls.Add(answersLabel);

panelMain.Controls.Add(questLabel);

panelMain.Controls.Add(panel1);

Program.data.numQuest++;

}

db.closeConnection();

}

private void answersLabel\_TextChanged(object sender,EventArgs e) {

TextBox TextBox=(TextBox)sender;

foreach (string line in TextBox.Lines) {

if (line.Length > 60) {

TextBox.Undo();

}

}

TextBox.ClearUndo();

}

private void button\_danger\_Click(object sender,EventArgs e) {

start.ActiveForm.Hide();

welcome welcome=new welcome();

welcome.ShowDialog();

Close();

}

private void button\_primary\_Click(object sender,EventArgs e) {

dialog\_newTest dialog\_newTest=new dialog\_newTest();

dialog\_newTest.ShowDialog();

this.testsTableAdapter.Fill(this.practic\_testDataSet.tests);

}

private void button\_success\_Click(object sender,EventArgs e) {

db.openConnection();

Program.data.idTest=comboBox1.SelectedValue.ToString();

MySqlCommand delTest=new MySqlCommand("DELETE FROM `questions` WHERE id\_test=@idtest;",db.getConnection());

delTest.Parameters.Add("@idtest",MySqlDbType.Int32).Value=Program.data.idTest;

try {

delTest.ExecuteNonQuery();

}

catch {

Program.err(res.err\_dbNotQuery);

}

int i=0;

int numQuest=1;

int numQuestDB=0;

char ans;

while (i < panelMain.Controls.Count) {

MySqlCommand insertQuest=new MySqlCommand("INSERT INTO `questions` (`id\_test`,`question`) VALUES (@idtest,@quest);",db.getConnection());

insertQuest.Parameters.Add("@idtest",MySqlDbType.Int32).Value=Program.data.idTest;

insertQuest.Parameters.Add("@quest",MySqlDbType.TinyText).Value=(panelMain.Controls[i+1] as TextBox).Text;

try {

insertQuest.ExecuteNonQuery();

}

catch {

Program.err(res.err\_dbNotQuery);

}

MySqlCommand lastId=new MySqlCommand("SELECT LAST\_INSERT\_ID();",db.getConnection());

try {

using (MySqlDataReader readerLastId=lastId.ExecuteReader()) {

if (readerLastId.Read()) {

numQuestDB=int.Parse(readerLastId[0].ToString());

}

}

for (int a=0; a < ((panelMain.Controls[i] as TextBox).Lines.Length); a++) {

MySqlCommand addAnswer=new MySqlCommand("INSERT INTO `variants` (`id\_question`,`variant`,`answer`) VALUES (@idquest,@variant,@answer);",db.getConnection());

addAnswer.Parameters.Add("@idquest",MySqlDbType.Int32).Value=numQuestDB;

addAnswer.Parameters.Add("@variant",MySqlDbType.VarChar).Value=(panelMain.Controls[i] as TextBox).Lines[a].ToString();

if (((panelMain.Controls[i+2] as Panel).Controls[a] as RadioButton).Checked) {

addAnswer.Parameters.Add("@answer",MySqlDbType.VarChar).Value="t";

ans='+';

}

else {

addAnswer.Parameters.Add("@answer",MySqlDbType.VarChar).Value="f";

ans=' ';

}

try {

addAnswer.ExecuteNonQuery();

}

catch (Exception ex) {

Program.err(ex.Message);

Program.err(res.err\_dbNotQuery);

}

}

}

catch (Exception ex) {

Program.err(res.err\_dbNotQuery);

}

i += 3;

numQuest++;

}

db.closeConnection();

}

}

}