Geekbrains

Специальность Программист «Цифровые профессии»

Дипломный проект

«Разработка бизнес приложение, для ведения оперативного учёта на предприятиях малого и среднего бизнеса.

Матвиенок Владимир Сергеевич

Г. Краснодар, 2024 год

**Дипломный проект**

**«Разработка бизнес приложение АИС Бизнес, позволяющее вести оперативный учёт на предприятиях малого и среднего бизнеса.**

Содержание

1. Введение.
2. Бизнес приложение АИС Бизнес, и способы получения данных.
3. Техническое задание для разработки бизнес приложения АИС Бизнес.
   1. Этап 1. Разработка десктопного бизнес приложения АИС Бизнес.
   2. Этап 2. Разработка Web-интерфейса бизнес приложения АИС Бизнес.
   3. Этап 3. Разработка мобильной версии бизнес приложения АИС Бизнес.
4. Способы хранения информации и передача обновлений на сторону заказчика.
5. Среда разработки и инструменты для разработки.
6. Выбор языка программирования для разработки приложения АИС Бизнес.
7. Практическая разработка бизнес приложения АИС Бизнес.
8. Тестирование бизнес приложения АИС Бизнес.
9. Ввод в опытную эксплуатацию бизнес приложения АИС Бизнес.
10. **Введение.**

Да, за последние 10 лет количество платформ увеличилось в десятки раз, и это стало следствием динамичного развития рынка IT, мобильных устройств и технологий. Если 12 лет назад фактически единственной платформой была Windows XP, то сейчас разработчикам софта нужны решения для Windows 10, Windows 8, macOS, Android (разных версий), iOS, веб-платформы и прочих систем — всего не перечислишь. И дело не только в софтверной платформе. Разнообразие устройств – телефоны, планшеты, ноутбуки, на которых это все крутится с разными размерами экрана, процессорами и объёмами памяти, — делает этот рынок ещё более многообразным.

Вне всяких сомнений, для пользовательской аудитории многообразие ОС и возможность выбора — это огромный плюс. Конечно, есть неудобства, связанные с переездом с одной платформы на другую, переносом данных, привыканием к новым интерфейсным решениям. Но это всё мелочи на фоне тех возможностей, которые предлагают современные приложения и которые появились в результате инноваций. Инновации — это фундаментальная вещь в сфере IT, именно они двигают рынок. До тех пор пока они существуют, пока кто-то придумывает новые, интересные, полезные и классные вещи, никакой унификации не будет. Унификация является тупиком и означает прекращение инноваций, а в индустрии высоких технологий это недопустимо. Да, нужны стандарты в некоторых “фундаментальных” областях, которые помогают всей индустрии, – например, TCP/IP, USB, HTML5. Но в большинстве случаев унификация – это попытка владельца платформы или технологии заставить всех этим пользоваться и собирать деньги.

Усложняет ли обилие программных платформ работу разработчикам? Безусловно. Но с другой стороны, многообразие ОС открывает массу новых возможностей для производителей софта и позволяет им получить доступ к широкой аудитории пользователей. Если вы разработчик какого-то приложения, то вам нужно делать версию своего продукта не только, к примеру, для iOS, но и для альтернативных платформ, будь то Android, Windows, Linux или Веб. При этом необходимо принимать во внимание форм-фактор устройств, на которых будет запускаться разрабатываемая программа, учитывать размер экрана и другие важные моменты. Это с одной стороны, а с другой — это просто цена, которую платишь, чтобы получить доступ к огромному количеству новых рынков и новых возможностей.

В последнее время софтверный рынок делает всё больший крен в сторону мобильных и облачных технологий. Не приведёт ли такая тенденция к исчезновению «классических» десктопных приложений с привычными для пользователя интерфейсами и инструментами работы с данными?

Это вопрос из серии: а исчезнут ли нативные приложения для iOS, прикажет ли долго жить App Store потому, что все перейдут на веб-приложения? Многие почему-то уверены, что дни десктопных программ сочтены, но это не так: просто сам рынок десктопов сокращается. Таких приложений миллионы, и они будут развиваться ещё много десятков лет, пока не появится какая-то другая платформа. Почему? Да потому, что ими удобнее пользоваться, они быстро работают и более функциональны, нежели мобильные и веб-приложения. Плюс нельзя списывать со счетов корпоративный рынок и бизнес-аудиторию, которые очень инертны и ещё долгое время будут отдавать предпочтение «классическим» приложениям. Крупные производители софта это понимают и продолжают в линейках своих продуктов развивать и поддерживать версии для стационарных рабочих станций. У Microsoft, к примеру, есть веб-приложения для Excel, Word, PowerPoint, которые могут запускаться с любого устройства, но наряду с ними компании предлагают своим пользователям родные клиенты для ПК, которые, конечно, работают быстрее и предоставляют больший набор функций, нежели использующие браузерное окружение веб-программы.

Веб-приложения получили путёвку в жизнь с утверждением консорциумом W3C стандарта HTML5, который сподвигнул многих серьёзных разработчиков к переводу своих приложений в онлайн. С одной стороны, HTML5 предоставил программистам много новых возможностей, с другой — заложенный в него потенциал не безграничен и уступает нативным приложениям. Конечно, в будущем получат своё развитие версии стандарта HTML 5.х, появятся новые редакции HTML 6.0, 7.0 и так далее. Но этот прогресс займёт очень продолжительное время. И ещё долго многие веб-версии приложений будут уступать родным аналогам этих приложений. Когда он произойдёт — сказать сложно.

По сути, с появлением веб-приложений и онлайновых сервисов началось возрождение тонких клиентов. Быстрый Интернет сейчас распространён повсеместно, размещённые в облаке серверные мощности — огромные, да и производительность смартфонов сейчас выше, чем у компьютеров 10 лет назад. Проблема, на мой взгляд, тут скорее не в мощностях, а в удобстве управления веб-приложениями и обеспечении их надёжной безопасности.

1. **Бизнес приложение АИС Бизнес, и способы получения данных.**

АИС Бизнес, далее - Автоматизированная Информационная Система Бизнес, предназначенная для:

- ведения оперативного учёта денежных средств, денежных потоков, источников финансирования, как отдельных инвестиционных проектов компании, так и финансово хозяйственной деятельности в целом.

- сбор данных о движении денежных средств предприятия, по средствам занесения данных через визуальные формы занесения данных.

- формирования отчётов о движении денежных средств компании.

- Работа с бюджетами компании. Составления Бюджета движения денежных средств (далее – БДДС), Бюджета Доходов и расходов (далее – БДР).

- Расчёт показателей работы компании, на основе полученных данных.

Получение данных должно осуществляться в ручном режиме. Занесения данных пользователями. А так же загрузки в автоматическом режиме с других учётных систем и сервисов.

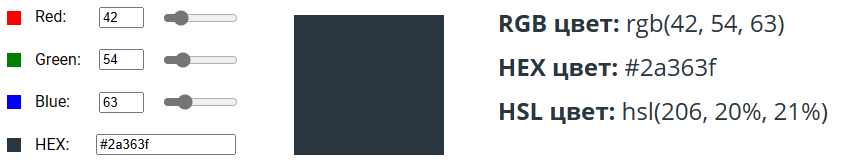
1. **Техническое задание для разработки приложения АИС Бизнес.**

Разработка приложения АИС Бизнес, в соответствии с требованиями заказчика разделена на 3(три) этапа.

Этап 1. Разработка десктопной версии приложения АИС Бизнес.

За основу, было принято, что разработка приложения, для операционной системы Windows, так как основная часть сотрудников компании базируется в центральном офисе компании, также специалисты ведут деятельность в структурных подразделениях компании. При принятии решения, о разработка приложения, для операционной системы Windows, послужило наличие собственного сервера компании находящегося на территории центрального офиса.

Визуальный вид приложения должен быть в корпоративных цветах компании.

Фон

Цвет кнопок

Шрифт - Franklin Gothic Book;

Размер шрифта - 9pt;

Вход в приложение должен осуществляться по логину и паролю, который пользователь указывает при регистрации.

Необходимо предусмотреть формы:

- Регистрации;

- Ввода пароля;

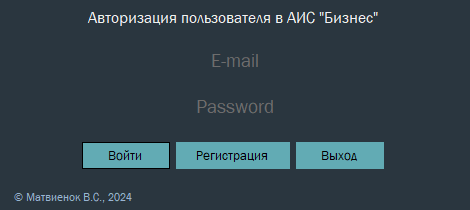
- Справочник организаций;

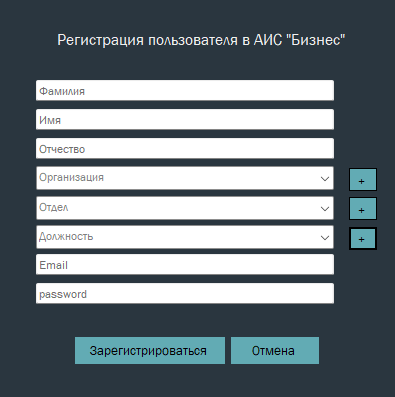
- Справочник должностей;

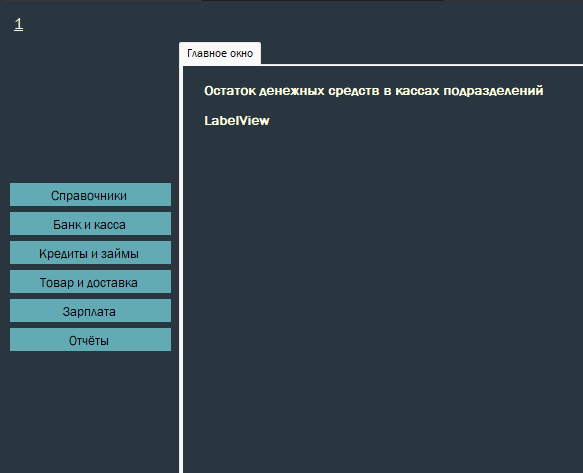
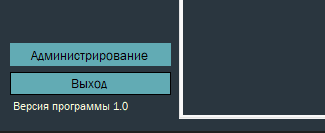
- Справочник подразделений организаций;

- Справочник сотрудников.

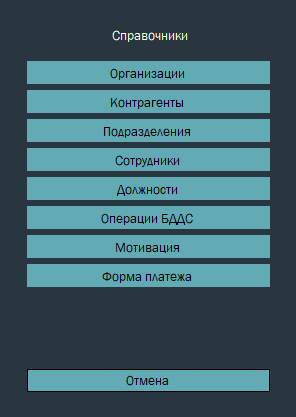
Как видит пользователь визуализацию приложения.

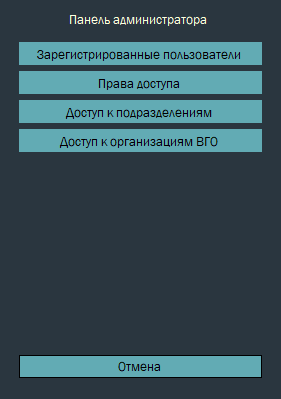
Форма авторизации пользователя:

Для незарегистрированных пользователей необходимо пройти регистрацию.

После регистрации и ввода учетных данных попадаем в главное окно программы.

Далее уже перемещаясь по меню попадаем в нужную вкладку.



Доступ к информации осуществляется в зависимости от роли зарегистрированного пользователя в системе. Доступ может назначить администратор системы, так же используя визуальные формы для взаимодействия с системой.

Этап 2. Разработка Web-интерфейса бизнес приложения АИС Бизнес.

Веб-приложение — это тип программного приложения, доступ к которому и использование которого осуществляется через веб-браузер. В отличие от традиционных настольных приложений, веб-приложения не нужно устанавливать на компьютер или устройство пользователя, и они предназначены для работы на любом устройстве, подключенным к Интернету.

Разработка веб-приложения будет осуществляться на Django.

Django - это фреймворк для создания веб-приложений с помощью языка программирования Python.

Django был создан в 2005 году, когда веб-разработчики из газеты Lawrence Journal-World стали использовать Python в качестве языка для создания веб-сайтов. А в 2008 году вышел публичный первый релиз фреймворка. На сегодняшний день он продолжает развиваться. Так, текущей версией фреймворка на момент написания этой статьи является версия 5.0, которая вышла в декабре 2023 года. Каждый новый релиз фреймворка выходит в средним каждые 8 месяцев. Кроме того, постоянно выходят обновления и исправления в безопасности.

Django довольно популярен. Он используется на многих сайтах, в том числе таких, как Pinterest, PBS, Instagram, BitBucket, Washington Times, Mozilla и многих других.

Фреймворк является бесплатным. Он развивается как open source, его исходный код открыт, его можно найти репозитории на githube.

На Django можно создавать широкий диапазон веб-приложений: от небольших персональных сайтов до высоконагруженных сложных веб-сервисов.

Django по умолчанию предлагает готовую функциональность для ряда распространенных задач, например, систему аутентификации, генерацию карт сайта и т.д., благодаря чему нам можно не изобретать велосипед и достаточно взять уже готовые компоненты.

В Django большое внимание уделяется безопасности, благодаря чему фреймворк помогает разработчикам избежать многих распространенных проблем в системе безопасности, например, sql-инъекций.

Основные элементы паттерна:

**URL dispatcher:** при получение запроса на основании запрошенного адреса URL определяет, какой ресурс должен обрабатывать данный запрос.

**View:** получает запрос, обрабатывает его и отправляет в ответ пользователю некоторый ответ. Если для обработки запроса необходимо обращение к модели и базе данных, то View взаимодействует с ними. Для создания ответа может применять Template или шаблоны. В архитектуре MVC этому компоненту соответствуют контроллеры (но не представления).

**Model:** описывает данные, используемые в приложении. Отдельные классы, как правило, соответствуют таблицам в базе данных.

**Template:** представляет логику представления в виде сгенерированной разметки html. В MVC этому компоненту соответствует View, то есть представления.

Когда к приложению приходит запрос, то URL dispatcher определяет, с каким ресурсом сопоставляется данный запрос и передает этот запрос выбранному ресурсу. Ресурс фактически представляет функцию или View, который получает запрос и определенным образом обрабатывает его. В процессе обработки View может обращаться к моделям и базе данных, получать из нее данные, или, наоборот, сохранять в нее данные. Результат обработки запроса отправляется обратно, и этот результат пользователь видит в своем браузере. Как правило, результат обработки запроса представляет сгенерированный html-код, для генерации которого применяются шаблоны (Template).

Перед началом работы с Django понадобится установить интерпретатор Python. Существуют разные способы установки Django.

Пакетный менеджер pip.

Пакеты Django размещаются в центральном репозитории для большинства пакетов Python - Package Index (PyPI). И для установки из этого репозитория нам потребуется пакетный менеджер pip. Менеджер pip позволяет загружать пакеты и управлять ими. Обычно при установке python также устанавливается и менеджер pip. В этом случае мы можем проверить версию менеджера, выполнив в командной строке/терминале команду pip -V.

Установка виртуальной среды

Виртуальная среда или venv не является неотъемлимой частью разработки на Django. Однако ее рекомендуется использовать, так как она позволяет создать множество виртуальных сред Python на одной операционной системе. Благодаря виртуальной среде приложение может запускаться независимо от других приложений на Python.

В принципе можно запускать приложения на Django и без виртуальной среды. В этом случае все пакеты Django устанавливаются глобально. Однако что если после создания первого приложения выйдет новая версия Django? Если мы захотим использовать для второго проекта новую версию Django, то из-за глобальной установки пакетов придется обновлять первый проект, который использует старую версию. Это потребует некоторой дополнительной работы по обновлению, так как не всегда соблюдается обратная совместимость между пакетами. Если мы решим использовать для второго проекта старую версию, то мы лишиемся потенциальных преимуществ новой версии. И использование виртуальной среды как раз позволяет разграничить пакеты для каждого проекта.

Для работы с виртуальной средой в python применяется встроенный модуль venv.

Активация виртуальной среды

Для использования виртуальную среду надо активировать. И каждый раз, когда мы будем работать с проектом Django, связанную с ним виртуальную среду надо активировать. Например, активируем выше созданную среду, которая располагается в текущем каталоге в папке .venv. Процесс активации немного отличается в зависимости от операционной системы и от того, какие инструменты применяются. Так, в Windows можно использовать командную строку и PowerShell, но между ними есть отличия.

Этап 3. Разработка мобильной версии бизнес приложения АИС Бизнес.

Разработка мобильного приложения должна осуществляться на языке Puthon, при помощи сторонних библиотек. Было принято решение использовать библиотеку фреймворка Kivy.

Kivy используется для создания естественных пользовательских интерфейсов с преимуществами абстракции и модульности. Он отделяет разработчика от ненужных и сложных деталей и просто предоставляет простые API для выполнения сложных задач.

Внутреннее устройство kivy основано на двух принципах:

Абстракция. Разработчики фреймворка постарались абстрагировать многие типовые задачи, такие как открытие окна, отображение картинок и текста, воспроизведение звука, исправление орфографии и т. д. Это упрощает и расширяет интерфейс программирования (API). Кроме того, в работе Kivy используются основные провайдеры сценариев, роль которых играют собственные API конкретной платформы. Они позволяют использовать функциональные возможности ОС, повышая эффективность работы во фреймворке. Более того, собственные библиотеки ОС сокращают размер дистрибутива Kivy.

Модульность. Модули — это дополнительные программные компоненты, расширяющие функциональность Kivy. По своему принципу они схожи с расширениями для веб-браузеров. Помимо использования уже существующих модулей, разработчик может написать свои собственные. Модульность делает разработку приложений в Kivy более гибкой.

Ядро фреймворка включает наиболее часто используемые функции:

часы — применяются для планирования событий, могут быть как однократными, так и многократными;

кэш — используется для кэширования часто используемых событий и функций в целях повышения производительности;

распознавание жестов — позволяет обнаруживать и различать основной набор управляющих движений пользователя, таких как смахивание вверх, вниз или в стороны, раздвигание пальцами и т. д.;

характеристики — это собственные классы свойств, связывающих код виджета с описанием пользовательского интерфейса.

Преимущества Kivy

Кросс-платформенность. Фреймворк Kivy предназначен для использования на всех распространенных мобильных и десктопных устройствах под управлением операционных систем Windows, MacOS, Linux, iOS, Android, а также поддерживает одноплатные компьютеры Raspberry Pi. Кроме того, разработка приложений с его помощью осуществляется по принципу «Пишите код один раз и запускайте на любой платформе». То есть программные продукты, созданные с использованием этого фреймворка, будут работать на всех поддерживаемых устройствах. Такой подход максимально упрощает процесс разработки, так как не нужно создавать и тестировать различные версии приложений под разные платформы, а охват пользователей будет при этом максимальным.

Использование Python. Хотя этот язык программирования не был разработан специально для создания пользовательских интерфейсов, он подходит для решения этой задачи благодаря простому синтаксису и наличию большого числа библиотек. Кроме того, он хорошо адаптирован именно под разработку кросс-платформенных программных продуктов.

Наличие собственного языка. Помимо Python, во фреймворке Kivy используется встроенный промежуточный язык разметки пользовательских интерфейсов Kivy. Он позволяет декларативно создавать дерево виджетов и естественным образом связывать их свойства друг с другом или с обратными вызовами. С его помощью можно очень быстро разрабатывать прототипы и гибко вносить изменения в пользовательский интерфейс. Это также облегчает разделение логики и UI приложения.

Открытый исходный код. Скачать Kivy можно на официальном сайте проекта абсолютно бесплатно. Это на 100% свободное программное обеспечение, распространяемое по лицензии MIT. Оно доступно как профессионалам, так и разработчикам-любителям. С его помощью можно свободно создавать не только приложения для самостоятельного или учебного использования, но и полноценные коммерческие продукты. Особенно это преимущество важно для небольших стартапов или одиночных специалистов, так как позволяет вести разработку даже с минимальным бюджетом.

Недостатки Kivy.

Упрощенный дизайн виджетов. Из-за кросс-платформенности самого фреймворка и создаваемых с его помощью приложений разработчики сделали дизайн базовых виджетов максимально усредненным. Это делает создаваемые с их помощью интерфейсы достаточно посредственными и невыразительными в визуальном плане. Частично этот недостаток компенсируется тем, что облик и анимацию виджетов можно настраивать под дизайн конкретной платформы. Кроме того, есть возможность использовать библиотеки виджетов от сторонних разработчиков.

Недостаточная информационная поддержка. Вокруг фреймворка Kivy сложилось большое сообщество профессионалов и программистов-любителей. Разработчики фреймворка выложили на официальном сайте галерею реализованных проектов, которые можно использовать для изучения тех или иных технических решений. В то же время в интернете представлено довольно малое количество информационно-справочных материалов, инструкций и другой документации по Kivy, что ограничивает возможности пользователя в решении проблем, возникающих в процессе работы с данной библиотекой.

Фреймворк Kivy был разработан для максимально простого и быстрого создания десктопных и мобильных приложений и получил широкую популярность как среди любителей, так и у профессионалов. Этому способствуют такие его преимущества, как кросс-платформенность, возможность подключения сторонних библиотек, гибкие настройки виджетов, наличие родного языка разметки, открытый исходный код. Продуманная архитектура фреймворка позволяет разрабатывать UI с поддержкой традиционных и инновационных способов ввода, прорисовывать как типовые, так и уникальные запоминающиеся интерфейсы.

1. **Способы хранения информации и передача обновлений на сторону заказчика.**

Обновление и поддержка программного продукта это процесс улучшения, оптимизации и устранения дефектов программного обеспечения (ПО) после передачи в эксплуатацию. Сопровождение ПО — это одна из фаз жизненного цикла программного обеспечения, следующая за фазой передачи ПО в эксплуатацию. В ходе сопровождения в программу вносятся изменения, с тем, чтобы исправить обнаруженные в процессе использования дефекты и недоработки, а также для добавления новой функциональности, с целью повысить удобство использования и применимость ПО.

Сопровождение программного обеспечения – это процесс поддержки и обслуживания программного продукта после его разработки и внедрения. Оно включает в себя ряд действий и мероприятий, направленных на обеспечение надлежащей работы программы, исправление ошибок, добавление новых функций и улучшение производительности.

Сопровождение программного обеспечения является неотъемлемой частью жизненного цикла программного продукта. Оно начинается с момента его внедрения и продолжается до его вывода из эксплуатации. В процессе сопровождения осуществляется постоянный контроль и поддержка программы, чтобы она соответствовала требованиям пользователей и функционировала без сбоев.

Сопровождение программного обеспечения включает в себя следующие основные задачи:

Исправление ошибок и дефектов программы;

Добавление новых функций и возможностей;

Адаптация программы к изменениям в окружающей среде;

Оптимизация производительности и устранение узких мест;

Поддержка совместимости с новыми версиями операционных систем и другого программного обеспечения;

Обновление документации и руководств пользователя;

Обучение пользователей и технической поддержки.

Сопровождение программного обеспечения играет важную роль в обеспечении долгосрочной работоспособности программы и удовлетворении потребностей пользователей. Оно позволяет улучшить качество программного продукта, устранить недостатки и добавить новые функции, что способствует повышению эффективности и конкурентоспособности организации.

Сопровождение программного обеспечения – это процесс поддержки и развития программного продукта после его выпуска и внедрения. Оно необходимо для обеспечения его надежной и эффективной работы на протяжении всего жизненного цикла.

Основная цель сопровождения программного обеспечения – обеспечить его стабильную работу и удовлетворение потребностей пользователей. Вот несколько причин, почему сопровождение программного обеспечения является важным:

Устранение ошибок и дефектов.

В процессе эксплуатации программного продукта могут возникать ошибки и дефекты, которые могут привести к неправильной работе или сбоям системы. Сопровождение программного обеспечения позволяет выявлять и исправлять эти ошибки, обеспечивая стабильную и надежную работу программы.

Добавление новых функций и возможностей.

Сопровождение программного обеспечения также включает в себя добавление новых функций и возможностей, которые могут быть полезны для пользователей. Это может включать улучшение интерфейса, добавление новых алгоритмов или интеграцию с другими системами.

Поддержка совместимости.

Сопровождение программного обеспечения также включает в себя поддержку совместимости с новыми версиями операционных систем, библиотек и другого программного обеспечения. Это позволяет программе работать на новых платформах и использовать новые возможности, что важно для обеспечения ее актуальности и конкурентоспособности.

Обновление документации и руководств пользователя.

Сопровождение программного обеспечения включает в себя обновление документации и руководств пользователя. Это позволяет пользователям лучше понимать, как использовать программу, и упрощает процесс обучения новых пользователей.

Обучение пользователей и технической поддержки.

Сопровождение программного обеспечения также включает в себя обучение пользователей и предоставление технической поддержки. Это позволяет пользователям получить помощь и решить возникающие проблемы, что способствует удовлетворенности их потребностей.

Сопровождение программного обеспечения играет важную роль в обеспечении долгосрочной работоспособности программы и удовлетворении потребностей пользователей. Оно позволяет улучшить качество программного продукта, устранить недостатки и добавить новые функции, что способствует повышению эффективности и конкурентоспособности организации.

Методы и инструменты сопровождения программного обеспечения.

Сопровождение программного обеспечения включает в себя различные методы и инструменты, которые помогают обеспечить эффективность и качество процесса. Рассмотрим некоторые из них:

Использование систем управления версиями Системы управления версиями (Version Control Systems, VCS) позволяют отслеживать изменения в исходном коде программы и управлять ими. Они позволяют команде разработчиков работать над проектом одновременно, отслеживать изменения, вносить исправления и восстанавливать предыдущие версии кода при необходимости. Популярные системы управления версиями включают Git, SVN и Mercurial.

Использование инструментов для отладки и тестирования.

Отладка и тестирование являются важными этапами сопровождения программного обеспечения. Использование специальных инструментов для отладки позволяет выявить и исправить ошибки в коде. Тестирование программы с помощью автоматизированных тестов помогает проверить ее функциональность и надежность. Популярные инструменты для отладки и тестирования включают отладчики, юнит-тестирование и инструменты для автоматизации тестирования.

Использование систем отслеживания ошибок.

Системы отслеживания ошибок (Bug Tracking Systems) позволяют команде разработчиков отслеживать и управлять ошибками, которые обнаруживаются в программном обеспечении. Они позволяют создавать задачи для исправления ошибок, отслеживать их статус и приоритет, а также вести обсуждение и комментирование. Популярные системы отслеживания ошибок включают Jira, Bugzilla и Redmine.

Использование систем документирования.

Документирование является важной частью сопровождения программного обеспечения. Использование систем документирования позволяет создавать и поддерживать документацию, описывающую функциональность программы, ее архитектуру, инструкции по установке и использованию, а также другую полезную информацию. Популярные системы документирования включают Markdown, Sphinx и Doxygen.

Использование систем автоматической сборки и развертывания.

Системы автоматической сборки и развертывания (Continuous Integration and Deployment, CI/CD) позволяют автоматизировать процесс сборки, тестирования и развертывания программного обеспечения. Они позволяют автоматически собирать и тестировать код при каждом изменении, а также автоматически развертывать новые версии программы на серверах. Популярные системы CI/CD включают Jenkins, Travis CI и CircleCI.

Это лишь некоторые из методов и инструментов, которые могут быть использованы в процессе сопровождения программного обеспечения. Выбор конкретных методов и инструментов зависит от требований проекта и предпочтений команды разработчиков.

В данном случае, сопровождение приложения АИС Бизнес будет осуществляться через репозиторий Git.

Git — это распределенная система управления версиями, которая означает, что локальный клон проекта — это полный репозиторий управления версиями. Полнофункциональные локальные репозитории упрощают работу как в автономном, так и в удаленном режиме. Разработчики фиксируют свою работу локально, а затем синхронизируют копию репозитория с копией на сервере. Эта парадигма отличается от централизованных систем управления версиями, где клиенты должны синхронизировать код с сервером перед созданием новой версии кода.

Гибкость и популярность Git делают его отличным выбором для любой команды. Многие разработчики и выпускники колледжа уже знают, как использовать Git. Сообщество пользователей Git создало ресурсы для обучения разработчиков и популярности Git, что упрощает получение помощи при необходимости. Почти каждая среда разработки поддерживает Git и средства командной строки Git, реализованные в каждой основной операционной системе.

Базовые возможности Git.

При каждом сохранении работы Git создает фиксацию. Фиксация — это моментальный снимок всех файлов в определенный момент времени. Если файл не изменился с одной фиксации на следующую, Git использует ранее сохраненный файл. Эта конструкция отличается от других систем, которые хранят начальную версию файла и сохраняют запись разностных с течением времени.

Одновременная разработка.

Каждый имеет собственную локальную копию кода и может одновременно работать в собственных ветвях. Git работает в автономном режиме, так как практически каждая операция является локальной.

Более быстрые выпуски.

Ветви позволяют гибко и одновременно разрабатывать. Основная ветвь содержит стабильный высококачественный код, из которого вы выпускаете. Ветви компонентов содержат работу, которая объединяется в основную ветвь после завершения. Разделив ветвь выпуска от разработки, проще управлять стабильным кодом и отправлять обновления быстрее.

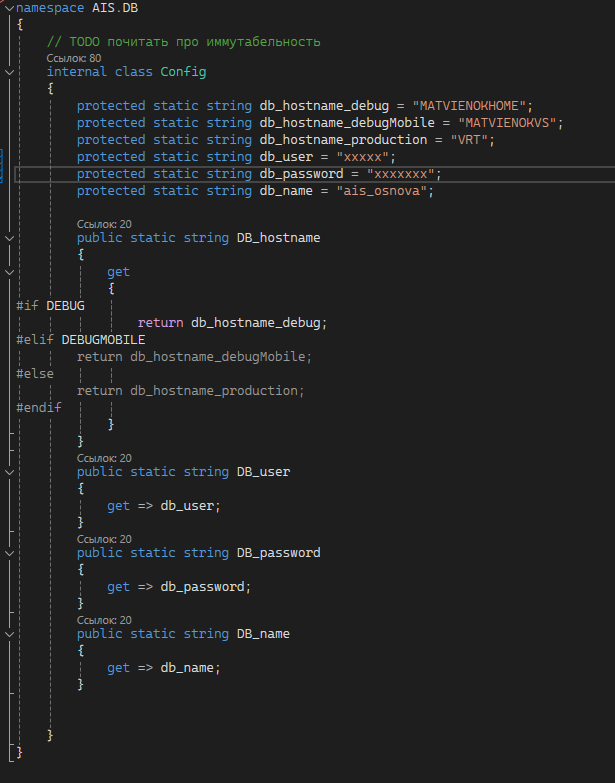
Встроенная интеграция.

Из-за своей популярности Git интегрируется в большинство инструментов и продуктов. Каждая основная интегрированная среда разработки поддерживает встроенную Git, а многие средства поддерживают непрерывную интеграцию, непрерывное развертывание, автоматическое тестирование, отслеживание рабочих элементов, метрики и интеграцию функций создания отчетов с Git. Эта интеграция упрощает повседневный рабочий процесс.

Файлы и фиксации.

Файлы в Git находятся в одном из трех состояний: изменено, поэтапно или зафиксировано. При первом изменении файла изменения существуют только в рабочем каталоге. Они еще не являются частью фиксации или истории разработки. Разработчик должен этапировать измененные файлы, которые должны быть включены в фиксацию. Промежуточная область содержит все изменения, которые необходимо включить в следующую фиксацию. После того как разработчик будет счастлив с промежуточными файлами, файлы упаковываются в виде фиксации с сообщением о том, что изменилось. Эта фиксация становится частью журнала разработки.

Для корректной работы приложения необходимо правильно настроить файл config, для взаимодействия с базой данных.

**В связи с тем, что проект является действующей разработкой, для предприятия среднего бизнеса, в целях обеспечения конфиденциальности, данные логина и пароля скрыты.**

Работа приложения осуществляется с базой данных MS SQL.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 2022, которая вышла в ноябре 2022 году и которая будет использоваться в текущем руководстве.

SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

SQL Server характеризуется такими особенностями как:

Производительность. SQL Server работает очень быстро.

Надежность и безопасность. SQL Server предоставляет шифрование данных.

Простота. С данной СУБД относительно легко работать и вести администрирование.

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД.

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Изначально язык SQL был разработан в компании IBM для системы баз данных, которая называлась System/R. При этом сам язык назывался SEQUEL (Structured English Query Language). Хотя в итоге ни база данных, ни сам язык не были впоследствии официально опубликованы, по традиции сам термин SQL нередко произносят как "сиквел".

В 1979 году компания Relational Software Inc. разработала первую систему управления баз данных, которая называлась Oracle и которая использовала язык SQL. В связи с успехом данного продукта компания была переименована в Oracle.

Впоследствии стали появляться другие системы баз данных, которые использовали SQL. В итоге в 1989 году Американский Национальный Институт Стандартов (ANSI) кодифицировал язык и опубликовал его первый стандарт. После этого стандарт периодически обновлялся и дополнялся. Последнее его обновление состоялось в 2011 году. Но несмотря на наличие стандарта нередко производители СУБД используют свои собственные реализации языка SQL, которые немного отличаются друг от друга.

Выделяются две разновидности языка SQL: PL-SQL и T-SQL. PL-SQL используется в таких СУБД как Oracle и MySQL. T-SQL (Transact-SQL) применяется в SQL Server. Собственно, поэтому в рамках текущего руководства будет рассматриваться именно T-SQL.

В зависимости от задачи, которую выполняет команда T-SQL, он может принадлежать к одному из следующих типов:

DDL (Data Definition Language / Язык определения данных). К этому типу относятся различные команды, которые создают базу данных, таблицы, индексы, хранимые процедуры и т.д. В общем определяют данные.

В частности, к этому типу мы можем отнести следующие команды:

CREATE: создает объекты базы данных

ALTER: изменяет объекты базы данных

DROP: удаляет объекты базы данных

TRUNCATE: удаляет все данные из таблиц

DML (Data Manipulation Language). К этому типу относят команды по выбору данных, их обновление, добавление, удаление - в общем все те команды, с помощью которыми мы можем управлять данными.

К этому типу относятся следующие команды:

SELECT: извлекает данные из БД

UPDATE: обновляет данные

INSERT: добавляет новые данные

DELETE: удаляет данные наиболее популярные версии Linux, Windows, MacOS.

1. **Среда разработки и инструменты для разработки.**

Существует множество сред разработки, которые различаются по функционалу и поддержке разных языков программирования. Вот некоторые из популярных сред:

Visual Studio — мультиязычная среда от Microsoft.

IntelliJ IDEA — среда разработки от компании JetBrains.

PyCharm — продукт компании JetBrains.

Eclipse — среда разработки, предназначенная специально для работы с языком программирования С.

Code::Blocks — гибкая IDE, одинаково хорошо работающая на разных платформах.

NetBeans — универсальная среда разработки с открытым кодом от Apache.

Komodo IDE — платное решение, но есть бесплатная версия для частного использования.

XCode — набор инструментов для программирования под iOS, iPadOS, macOS, watchOS и tvOS.

Для реализации данного проекта была выбрана Visual Studio.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки (IDE) программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, UWP а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone .NET Compact Framework и Silverlight. После покупки компании Xamarin корпорацией Microsoft появилась возможность разработки IOS и Android программ.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

1. **Выбор языка программирования для разработки приложения АИС Бизнес.**

Согласно техническому заданию, предоставленному заказчиком, язук, на котором должен быть исполнен 1 этап должен отвечать следующим требованиям.

- Достаточно популярный.

- Кросс – платформенный - запускается почти на любом железе.

- Постоянно развивающийся.

- Дружить с экосистемой Windows.

C# — это язык программирования от компании Microsoft. Изначально его создавали для проектов под Windows, но теперь это по-настоящему универсальный язык: на нём пишут игры, десктопные приложения, веб-сервисы, нейросети и даже графику для метавселенных.

С# был создан в 1993—2001 годах. Разработкой занималась группа инженеров из компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота. Предназначение С# — создание приложений для платформы Microsoft.

В нулевых годах компания разработала различные версии новых технологий и решений, использовавшихся для обмена сообщениями и данными. Кроме того, они применялись для создания веб-приложений. Кроме того, Microsoft сформировала платформу, которая позволила разрабатывать новые решения — .NET. Она включала в себя целый ряд языков программирования. Такой подход являлся очень необычным.

Платформа .NET имела ещё одно важное достоинство — технологию активных серверных страниц ASP.NET (Active Server Page). Она позволяла довольно быстро создавать веб-приложения, которые работали с базами данных. C# выпустили специально для ASP.NET (она также была написана на данном языке).

Используя объектно-ориентированный подход, программисты пишут на C# крупномасштабные, но, тем не менее, гибкие приложения, которые можно расширять по мере необходимости.

C# — относительно молодой язык. Однако он уже успел получить широкое распространение. Самая первая версия языка была выпущена вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Последней версией является C# 8.0. Она была выпущена в сентябре 2019 года вместе с релизом .NET Core 3. Преимущества C#.

Независимость от железа.

Программисту не надо адаптировать программу под разные платформы и системы — за него это делает виртуальная машина, вшитая в .NET Framework. В итоге один и тот же код можно запускать на любых устройствах — смартфонах, компьютерах, серверах, банкоматах и даже умных часах.

Отличная совместимость с Windows.

Не зря же язык разработали именно в Microsoft. Так же как Swift идеально подходит для программирования под экосистему Apple, C# прекрасно вписывается в экосистему Windows.

Управление памятью.

Чтобы программа работала стабильно, её надо иногда чистить от ненужных объектов, ссылок, кэша и прочего мусора. В C# это происходит автоматически — разработчику не надо следить за расходом памяти, бороться с её утечками или удалять мёртвые куски кода.

Строгая типизация.

Когда вы объявляете переменную в C#, надо сначала указать, что в ней лежит — строка, число или массив. Так разрабатывать чуть дольше, зато ваш код работает предсказуемо — числа взаимодействуют с числами, строки со строками и так далее. В языках со слабой типизацией свободы и драйва больше, но есть шанс пропустить ошибку, которая всплывёт в готовой программе.

Большое сообщество.

На С# пишут более миллиона программистов по всему миру. В соцсетях полно чатов и сообществ «шарпистов», где можно задать вопрос, обсудить сложную тему или найти готовое решение. В теории можно даже найти ментора, который поделится знаниями и поможет быстрее освоить язык.

Синтаксический сахар.

В С# есть много способов сократить код, не нарушая логику программы. Программисты называют такие приёмы «синтаксическим сахаром» — они помогают сделать код проще, понятнее и в целом симпатичнее. Сравните, например, как выглядит сложение чисел с «сахаром» и без.

Недостатки языка

Есть у C# и недостатки, причём довольно серьёзные.

Скорость. Когда мы запускаем программу на C#, код исполняется не сразу, а сначала адаптируется под нужное железо. Так мы охватываем больше платформ, но теряем в скорости — программе нужно сделать двойную работу, чтобы просто стартовать. Из-за этого интерфейсы на С# иногда подтормаживают при первом запуске.

Безопасность. Эксперты говорят, что код на C# легко декомпилировать — то есть перевести из машинного обратно в человеческий. Проблема в том, что так программу может легко прочитать хакер или конкурент — и изучить её уязвимости, украсть фрагменты кода или написать для неё вредоносный софт.

Мало доступа к железу. Так как С# — язык высокого уровня, на нём редко пишут проекты, где нужно полное взаимодействие с железом, — игровые движки, операционные системы, авиационный софт и так далее. Та же Unity целиком написана на низкоуровневом языке C++, хотя и умеет исполнять С#-команды.

ОС Windows почти целиком написана на С#. С помощью данного языка создано огромное количество популярных утилит и приложений: мессенджер Skype, браузер Internet Explorer, среда для разработки Visual Studio 2012, Microsoft Office (все его составляющие, включая Word, PowerPoint, Excel, Outlook и т.д.), продукты компании Adobe (Photoshop, Lightroom), браузер Mozilla Firefox и Winamp.

1. **Практическая разработка приложения.**

Разработка начата с установки среды разработки. Скачивание дистрибутива осуществляется с официального сайта Microsoft.

[Microsoft Visual Studio (scriptload.net)](https://visualstudio.scriptload.net/?yclid=17042428948943470591)

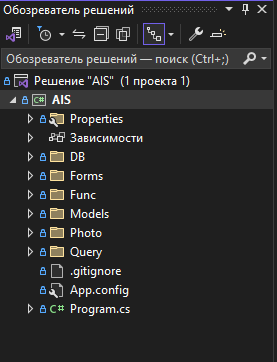
<https://visualstudio.scriptload.net/?yclid=17042428948943470591>

Visual Studio – это комплексная интегрированная среда разработки (IDE), которую можно использовать для написания, редактирования, отладки и сборки кода, а затем для публикации приложения. Помимо редактирования и отладки кода, Visual Studio включает графические конструкторы, компиляторы, средства завершения кода, систему управления версиями, расширения и многие другие функции для улучшения каждого этапа процесса разработки программного обеспечения.

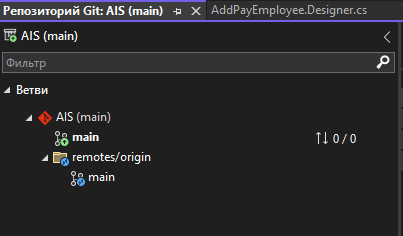
Также для выполнения задач, поставленных в техническом задании, понадобится, Система управления базами данных (далее – СУБД). Для хранения и накопления поступающей информации была выбрана СУБД от компании Microsoft, которая называется SQL Server Management studio. Которую так же можно скачать на официальном сайте.

[Скачивание SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Learn](https://learn.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16)

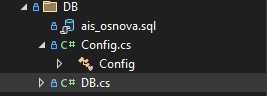
<https://learn.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>.

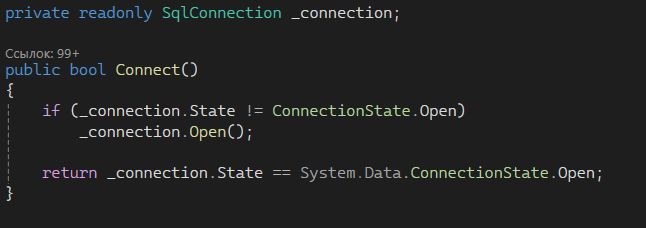
Далее необходимо произвести настройку программного обеспечения и создать новый проект. Решения и проекты можно просматривать в окне инструментов, которое называется обозревателем решений. На следующем снимке экрана показан пример консольного решения C# в Обозреватель решений, который содержит два проекта: Калькулятор и КалькуляторLibrary. Каждый проект содержит несколько файлов и папок. Полужирным шрифтом выделено имя запускаемого проекта. Этот проект выполняется при запуске приложения. Вы можете выбрать запускаемый проект.

Как можно увидеть из обозревателя решений, приложение состоит из основного набора папок, которые хранят в себе классы и компоненты, в которых пишется код программы.

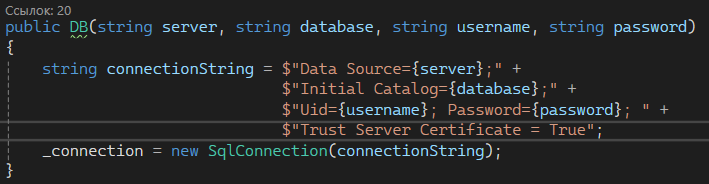
В целях сохранения данных, а так же предоставления возможности для совместной разработки, проект необходимо синхронизировать с Git. Это позволяет сделать непосредственно из среды разработки, либо через командную строку Power Shell 7.

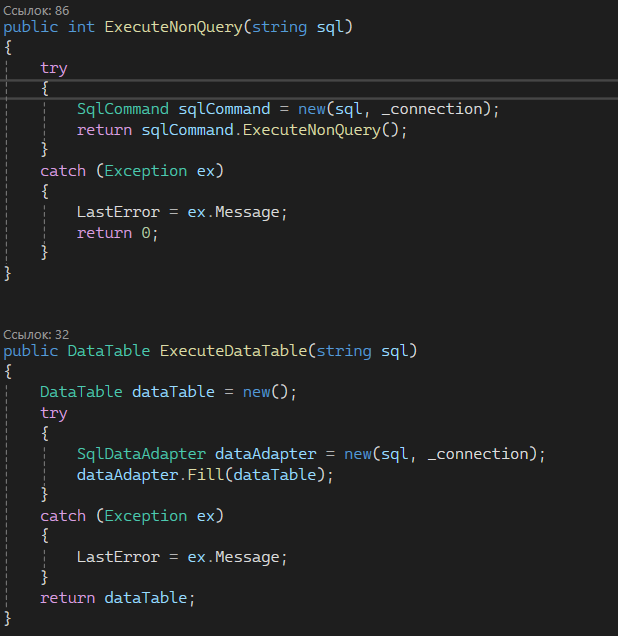
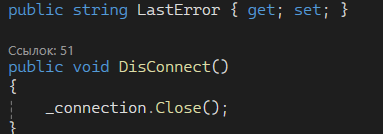
Работа с базой данных.

Для работы приложения необходимо настроить подключения к базе данных. Для этого создаём класс подключения к базе данных.



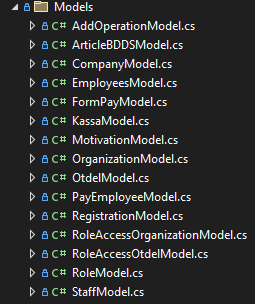
Все создаваемые классы должны отвечать требованию «Принципу единой ответственности, который говорит о том, что «Модуль должен отвечать за одного и только за одного актора.», другими словами, если модуль отвечает за взаимодействие с базой данных, то другого кода быть не должно.

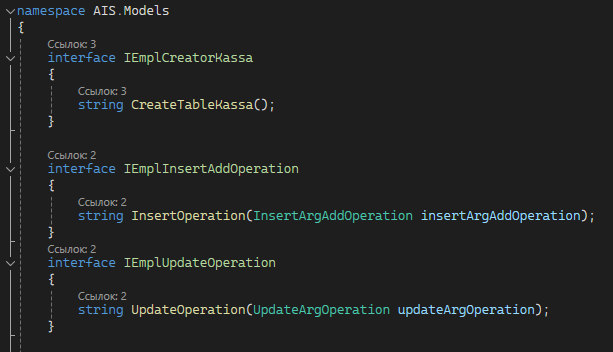
Поэтому, данный класс был расширен методами отвечающими за взаимодействие с базой данных в части выполнения запросов, получения информации из базы данных, а так же класс отвечающий за хранение строки подключения к базе данных.

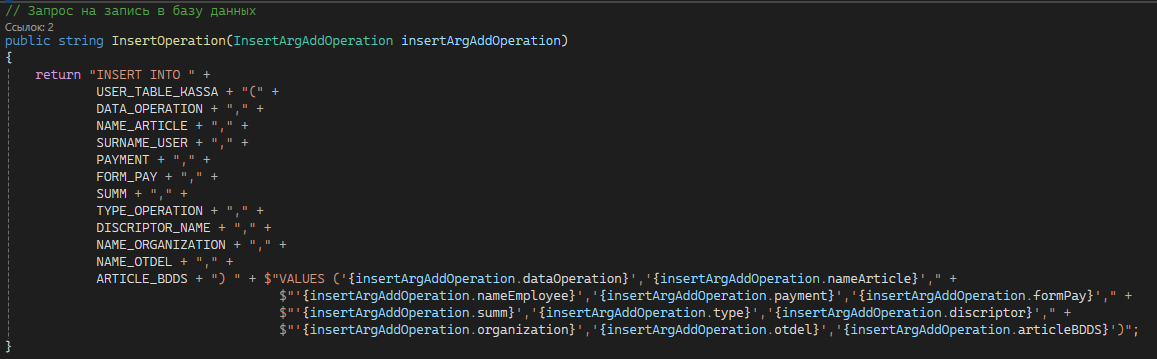
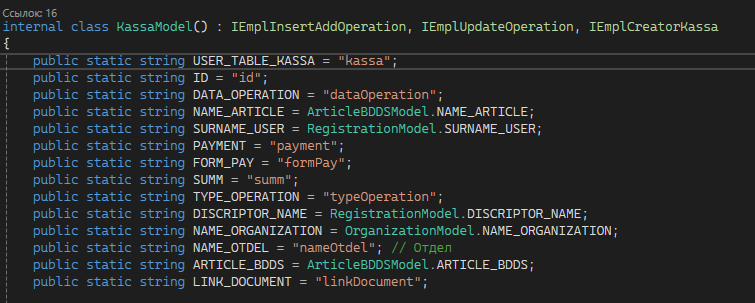
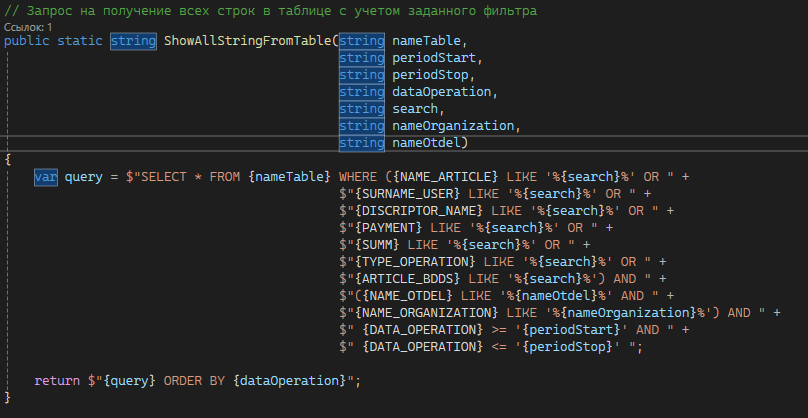


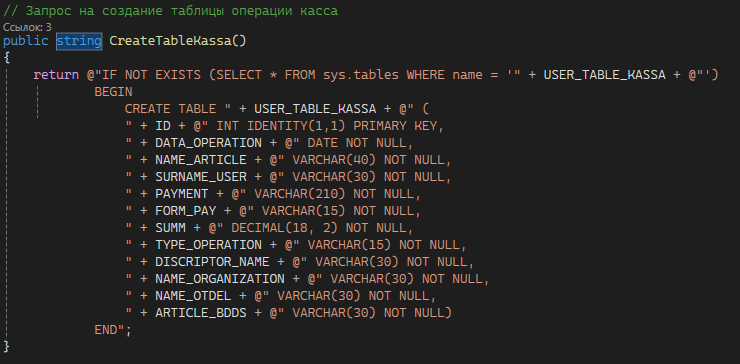
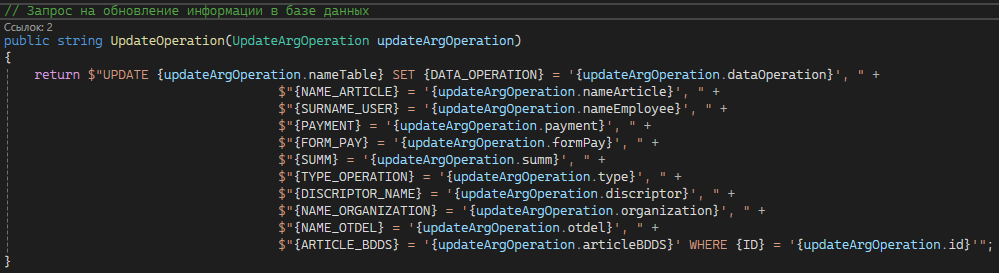
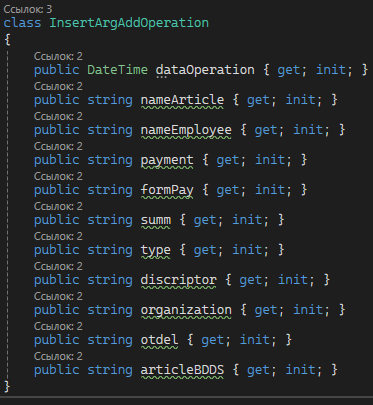
Работа с моделями данных.

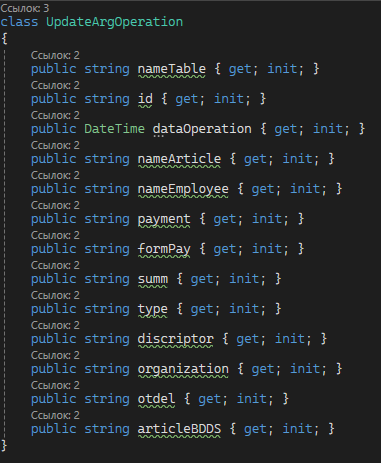
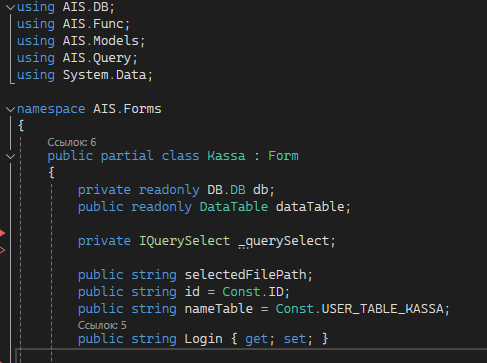
Модели представляют собой простые классы и располагаются в проекте в каталоге Models. Модели описывают логику данных.

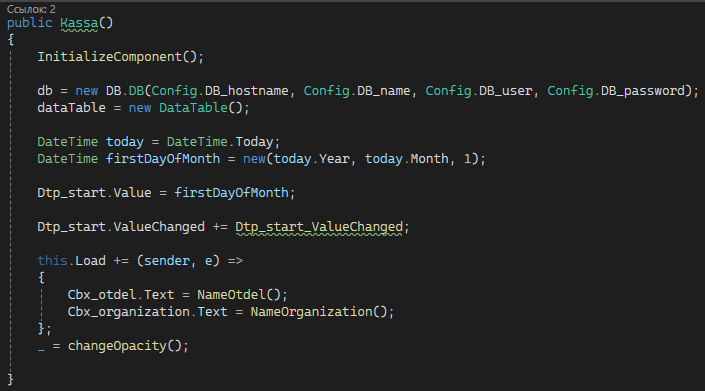
Для каждой визуальной формы в проекте содержится модель поведения и описания логики взаимодействия с базой данных, с соответствующими SQL запросами, константами, интерфейсами и классами.

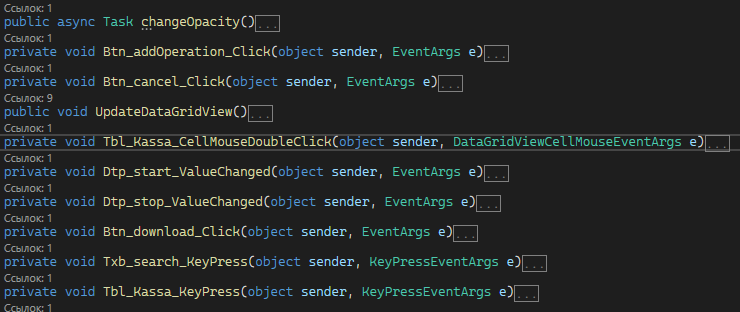
Все классы и модели разрабатываются по одной схеме, поэтому достаточно будет рассмотреть одну из них.

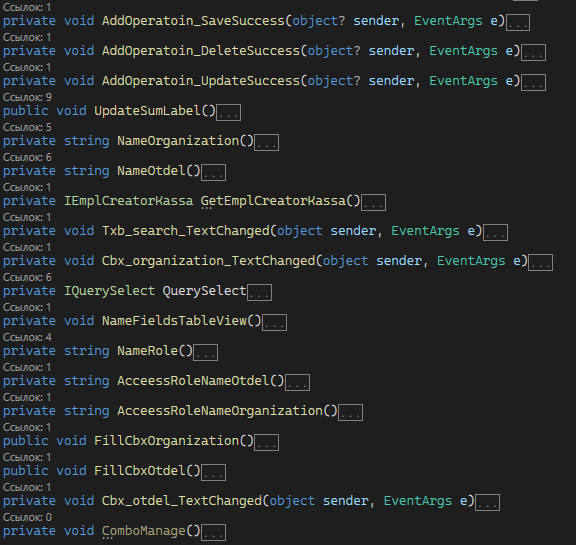
 



Переходя в основной класс, данной формы, что можно сказать. В нём содержится вся логика взаимодействия данного модуля с другими модулями и программой в целом.



Весь код максимально разбит на вспомогательные классы и отвечает требованиям SOLID.



1. **Тестирование приложения АИС Бизнес.**

Тестирование программного обеспечения играет важную роль в современном мире, где компьютерные программы проникают во все сферы нашей жизни. Без надлежащего тестирования программы могут быть подвержены сбоям, что в конечном итоге может привести к непредсказуемым последствиям и неудовлетворенности пользователей. В силу этого, тестирование является неотъемлемой частью разработки нового программного обеспечения, гарантирующей его качество, надежность и эффективность. Это процесс, позволяющий выявить и исправить проблемы, а также убедиться в соответствии новой программы требованиям и ожиданиям клиентов. В этой статье рассмотрим основные аспекты тестирования, важность его роли, типы и преимущества, которые оно предоставляет в области разработки программного обеспечения.

**Тестирование** — это процесс проверки программного обеспечения, системы или приложения на соответствие определенным требованиям и оценки их качества.

Оно выполняется с целью выявления ошибок, неполадок и нежелательного поведения программного продукта.

Определение слова “тестирование” имеет много значений. Рассмотрим основные:

Процесс выполнения программы с целью нахождения ошибок.

Интеллектуальная дисциплина, имеющая целью получение надежного программного обеспечения без излишних усилий на его проверку.

Проверка соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выполненных определенным образом.

Процесс наблюдения за выполнением программы в специальных условиях и вынесения на этой основе оценки каких-либо нюансов ее работы.

Процесс, имеющий целью выявление ситуаций, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и для определения поломок.

Тестирование программ является важной практикой по нескольким причинам:

Выявление ошибок. Позволяет обнаружить ошибки и недочеты в программном обеспечении. Раннее обнаружение и исправление ошибок способствует улучшению качества программы и уменьшению возможных проблем и рисков в дальнейшем.

Гарантия качества. Помогает проверить, насколько программа соответствует своим требованиям и спецификациям. Это позволяет удостовериться, что программа работает правильно, выполняет задачи и доставляет ожидаемые результаты.

Улучшение надежности. Способствует повышению надежности программного обеспечения. Через тестирование можно выявить уязвимости, ошибки в обработке данных и другие проблемы, которые могут привести к сбоям или неправильной работе программы.

Оптимизация производительности. Позволяет оценить производительность программы, выявить узкие места и бутылочные горлышки, которые могут замедлять работу программы.

Повышение удовлетворенности пользователей. Позволяет выявить и исправить проблемы, которые могут негативно влиять на пользовательский опыт. Корректная и надежная работа программы улучшает удовлетворенность пользователей и способствует их лояльности.

Уменьшение рисков и затрат. Помогает снизить риски, связанные с неправильной работой программы. Обнаружение и устранение ошибок на ранних стадиях разработки экономит время, усилия и ресурсы, которые могут быть затрачены на исправление проблем в более поздних этапах.

1. **Ввод в опытную эксплуатацию приложения АИС Бизнес.**

Развертывание системы на площадке опытной эксплуатации.

В соответствии с спроектированной технологической архитектурой, отраженной в документации, закупается серверное, коммуникационное и прочее оборудование, а также системное ПО. Компоненты информационной системы монтируются в единый программно-аппаратный комплекс, на площадках, на которых планируется его промышленное использование.

Ввиду того, что в крупных проектах, задействовано большое количество оборудования на котором ПО разбросано по нодам, узлам и даже облакам, то этот процесс необходимо сопровождать полномасштабным документированием. Например, в техдокументацию включаются таблицы с адресами серверов, рабочих мест, способов доступа и т.п. Для визуального представления используются диаграммы компонентов, дающие понимание расположения узлов сети, распределения компонентов их взаимодействия и т.п. А ведь еще должны быть определены мероприятия, регламентирующие всевозможные изменения в инфраструктуре, позволяющие устранить последствия отказов различных элементов системы.

Обучение персонала заказчика работе с информационной системой.

Как уже неоднократно упоминалось, в больших проектах особое внимание уделяется качеству документации, включая и инструкции пользователей системы. Чаще всего инструкции пользователей делятся на сегменты по видам деятельности, специализации и т.п. Это позволяет акцентировать внимание в документе на важные моменты и не грузить пользователей ненужной для них информацией.

Поскольку в обучении может быть задействовано значительное количество различных сотрудников заказчика, которые в свою очередь, для обеспечения непрерывности деловых процессов не могут обучаться в одно и то же время, которые должны обучаться различным функциональным обязанностям и по прочим уважительным причинам, необходимо тщательно планировать процесс подготовки персонала. Также полезно разбивать обучающихся на группы по категориям, требующим использование различных подходов и глубины обучения, исходя из уровня их первоначальной подготовленности. В итоге, составленный план-график обучения должен быть согласован со всеми заинтересованными лицами, и утвержден руководством заказчика, как обязательный к исполнению.

Выявление недостатков и дефектов информационной системы.

Очень часто в больших проектах, тестирование финального релиза не позволяет выявить все проблемные места решения. Причиной тому могут быть: огромные объемы данных на деле в «боевых» условиях, проявление уникальных сочетаний бизнес правил в реальных деловых процессах, особенности работы конкретного оборудования, специфические сочетания компонентов системы, балансирование нагрузки между распределенными узлами и т.п.

Зачастую ситуация еще осложняется тем, что внедрение новых систем на начальных стадиях ни в коей мере не отменяет необходимость производить работы на старых системах. То есть пользователи дублируют данные в обоих системах. Иногда требуется миграция существующих актуальных данных из устаревших хранилищ в новые, а структура и формат информации обычно весьма и весьма отличаются. Например, если в новой структуре данных не хватает информации для заполнения обязательных реквизитов, они заполняются какими-то данными назначенными «по умолчанию», а потом уже корректируются вручную пользователями. И это только малая толика того, с чем приходится сталкиваться в реальных проектах.

Отдельная тема — интеграционные решения, в которых может происходить сбои в цепочке, использующей различные компоненты, разработанные двумя, тремя и больше командами. Найти виноватых в этой ситуации крайне сложно, поскольку дефекты чаще всего возникают на стыке интеграционных элементов, из-за выявленных в ходе внедрения несоответствий. И тут важно не искать виновных для наказания, а быстро и конструктивно договориться о совместных уступках разработчиков стыкуемых компонентов, и эффективно решить проблему.

Учитывая все вышеперечисленное, этап опытной эксплуатации, чаще всего, насыщен эмоциональными всплесками и взаимными претензиями, как между командами разработчиков, так и с заказчиками. В этом случае очень важна роль архитекторов и системных аналитиков, которые должны оперативно локализовать проблему, предложить ее решение и согласовать его со всеми заинтересованными лицами. Для выполнения подобных работ требуется помимо основных профессиональных навыков, еще и обладание талантом переговорщика, и знанием основ менеджмента.

Согласование изменений в процессе внедрения информационной системы

Если работа некоторых функциональных модулей информационной системы критически не соответствует потребностям и ожиданиям заказчика, и найдены решения по преодолению этих проблем, то необходимо их зафиксировать и согласовать с заказчиком.

Этап согласования нового решения очень важен, как минимум по двум причинам.

Во-первых, если объем реализации изменений превысит суммы, заложенные на подобные риски в плане проекта, то необходимо либо заключать дополнительные соглашения, либо команда исполнителей будет работать в убыток. Зачастую исполнителей призывают по-быстрому сделать изменения, а мол учтем их и рассчитаемся за работы по ним потом, одним пакетом. Но по факту же такие случаи, обычно приводят, к тому, что заказчик опосля напрочь забывает свои обещания, а выполненные работы объявляет — исправлением исполнителями своих собственных ошибок.

Во-вторых, любые изменения одних компонентов системы могут повлечь за собой неизбежное изменение взаимозависимых компонентов, что требует тщательного анализа и, возможно, перепроектирования целой цепочки подсистем. В противном случае неизбежно возникновение дефектов в работе системы в целом. Проявляться это может например, в отказе работы модуля смежной команды исполнителей, и заказчик уже их объявляет халтурщиками и бракоделами.

Доработка информационной системы по итогам опытной эксплуатации.

Если в ходе опытной эксплуатации принимаются и согласуются решения о внесении изменений в разработанный программно-аппаратный комплекс, то на основании их выставляются задачи исполнителям по их реализации. Процесс, описанный в разделе Часть 3. Реализация проектного решения повторяется.

Если на стадии проектирования системы мы обсуждали отрицательное влияние полномасштабного использования методологии Scrum (1) в больших проектах, то на данном этапе она подходит как нельзя лучше. Особенно это ощутимо в проектах в которых продукт, переданный заказчику, не устраивает его по большей части показателей. Иными словами, пора поддаться панике и очень быстро, «сломя голову» вносить изменения в продукт, который уже эксплуатируют.

Собственно говоря, наступил момент, когда актуальны следующие условия:

Заказчик уже начал реально работать с системой, у него для этого выделено время, и он теперь наглядно представляет, что же ему действительно необходимо. Соответственно он готов плотно работать с командой исполнителей и у него есть в этом критическая необходимость;

Документация большей частью уже готова и ее изменение и дополнение может вестись уже не так оперативно, а оформляться постфактум по результатам успешной реализации.

Доработки большей частью происходят в отдельных модулях, подсистемах, контурах, у которых есть конкретная команда исполнителей, отвечающая за сегмент. Поэтому общение пользователей с разработчиками уже локализовано, легко установить качественную обратную связь;

Доработки и исправления необходимо выполнять очень оперативно, небольшими очередями с передачей результата заказчику, который в них кровно заинтересован;

Передача информационной системы в промышленную эксплуатацию.

Когда в ходе опытной эксплуатации решены все спорные вопросы и недоразумения по поводу того как должна функционировать внедренная система, и насколько она соответствует договору на ее разработку, стороны подписывают акты о выполнении контракта. Заказчик осуществляет полный расчет за выполненные работы. Договор на разработку и внедрение информационной системы может считаться выполненным.

Внедрение переходит в фазу промышленной эксплуатации. Эти взаимоотношения чаще всего юридически регулируются уже отдельным договором или дополнительным соглашением на сопровождение промышленной эксплуатацией системы. В рамках этого контракта могут происходить профилактические работы по диагностике работы компонентов системы, их взаимодействия, устранение мелких сбоев и т.п.

Этап внедрения информационной системы, являет собой момент истины всего процесса ее производства и ознаменует старт самого тяжелого периода для всех участников проекта. Он может включать в себя следующие активности:

Развертывание системы на площадке промышленной эксплуатации, включая поставку оборудования, установку системного ПО, установку актуального релиза внедряемой системы и т.п.;

Обучение пользователей работе с системой, включая администраторов, специалистов по обслуживанию оборудования и т.п.

Выявление и устранение недостатков и дефектов, выявленных в ходе опытной эксплуатации.

Согласование изменений в работе системы и приведение ее к соответствию контрактным обязательствам;

Подписание документов о выполнении договорных обязательств. Произведение полного расчета за выполненные работы;

Ввод системы в промышленную эксплуатацию;

**Список литературы:**

1. Алан Болье «Изучаем SQL»
2. Аллен Тейлор «SQL для чайников»
3. Программирование на Python в примерах и задачах. Васильев А. Н.
4. Простой Python. Современный стиль программирования. 2-е изд. Любанович Б.
5. Изучаем программирование на Python. Бэрри Пол.
6. C# 10 и .NET 6. Современная кросс-платформенная разработка. Марк Прайс
7. C# на примерах. Практика, практика и только практика. Евдокимов П В.