Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

РАЗВЕРТКА ВНУТРЕННЕЙ ИНФРАСТРУКТУРЫ РАЗРАБОТКИ

Отчет по лабораторной работе №1 по дисциплине «Новые технологии в программировании»

Студент гр. 588-1

_____/ Колесников А.М.
(подпись)
« 22 » февраля 2021г.

Руководитель старший научный сотрудник, доцент каф. КСУП

_____ / Горяинов А.Е.
(подпись)
«___» _____20__г.

Томск 2021

Оглавление

Введение

Цель работы: изучить пакет программ, используемых при разработке декстоп-приложений и получить умения их развертки на рабочей машине.

Задачи:

1. Ознакомиться с перечнем программ, используемых при разработке декстоп-приложений
2. Установить требуемые приложения
3. Изучить принцип работы с системами версионного контроля и модели ветвления при командной и индивидуальной разработке
4. Создать репозиторий проекта
5. Создать решение в репозитории

2 Программы и сервисы для разработки приложения

Инфраструктура разработки ПО – набор программных и аппаратных средств, а также правила взаимодействия в команде, обеспечивающих процесс разработки ПО. Как правило, в инфраструктуру разработки входят: компьютеры разработчиков, сервер/серверы; программное обеспечение для написания кода, проектирования, тестирования, дизайна и макетирования и т.д.; перечень должностей в команде, их должностные обязанности и регламент взаимодействия участников разработки между собой – для поддержания дисциплины и контроля выполнения хода работы.

Ключевые этапы разработки:

1) Определение проблемы и постановка задачи.

2) Составление технического задания, планирование сроков и ресурсов на разработку.

3) Макетирование (разработка пользовательского интерфейса, прототипирование) – определение внешнего вида будущей системы.

4) Проектирование - определение всех необходимых внутренних компонентов будущей системы.

5) Разработка – реализация внешних и внутренних компонентов согласно макетам и проектной документации.

6) Тестирование – проверка соответствия разработанной системы исходному техническому заданию

7) Внедрение – передача готовой системы заказчику или конечным пользователям

В ходе данной лабораторной работы предполагается развернуть минимально необходимую инфраструктуру для дальнейшей разработки ПО. В

ходе выполнения лабораторных работ потребуются следующие программы и сервисы:

1) Среда разработки: Microsoft Visual Studio 2017 Community.

2) Вспомогательные плагины для среды разработки: JetBrains Resharper.

3) Система версионного контроля: git с использованием сервиса GitHub.com.

4) Сборка установочных пакетов: InnoSetup.

5) Microsoft Word 2016.

6) Создание диаграмм технической документации: Spark Enterprise Architect

Microsoft Visual Studio 2018 Community – бесплатная среда разработки. Несмотря на громоздкость по сравнению с аналогами, данная среда содержит визуальные дизайнеры, которые в значительной степени упрощают верстку пользовательских интерфейсов приложений.

JetBrains Resharper – платная утилита, устанавливаемая поверх Visual Studio, содержащая полезные инструменты для написания кода. Утилита в реальном времени анализирует написанный код, находит в нём ошибки и предлагает варианты их исправления.

GitHub.com – веб-сервис, позволяющий хранить промежуточные версии исходного кода. Таким образом, в случае написания непоправимых ошибок в исходном коде или его потери, сервис позволит восстановить проект. Также является обязательным инструментом для организации командной работы, используемым во многих IT-компаниях.

InnoSetup – бесплатное и относительно простое приложение для создания установочных пакетов для десктоп-приложений.

Enterprise Architect – платное приложение, специализированное для создания диаграмм в области разработки ПО. Содержит готовые графические элементы для создания большинства видов программных диаграмм, в том числе и в нотации UML.

3 Версионный контроль

Система управления версиями (СУВ) — это система, сохраняющая изменения в одном или нескольких файлах так, чтобы потом можно было восстановить определённые старые версии. Для примеров в этой книге мы будем использовать исходные коды программ, но на самом деле можно управлять версиями практически любых типов файлов.

Виды систем управления версиями:

1) Локальные СУВ с простой базой данных, в которой хранятся все изменения нужных файлов. Одной из наиболее популярных СУВ данного типа является rcs, которая до сих пор устанавливается на многие компьютеры.

2) Централизованные СУВ. В таких системах, например CVS, Subversion и Perforce, есть центральный сервер, на котором хранятся все отслеживаемые файлы, и ряд клиентов, которые получают копии файлов из него.

3) Распределенные СУВ. В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто забирают последние версии файлов, а полностью копируют репозиторий. Поэтому в случае, когда по той или иной причине отключается сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных.

Почти все операции в Git – локальные. Для совершения большинства операций в Git необходимы только локальные файлы и ресурсы, т.е. обычно информация с других компьютеров в сети не нужна.

Git следит за целостностью данных. Перед сохранением любого файла Git вычисляет контрольную сумму, и она становится индексом этого файла. Поэтому невозможно изменить содержимое файла или каталога так, чтобы Git не узнал об этом. Эта функциональность встроена в сам фундамент Git и

является важной составляющей его философии. Если информация потеряется при передаче или повредится на диске, Git всегда это выявит.

Чаще всего данные в Git только добавляются. Практически все действия, которые совершаются в Git, только добавляют данные в базу. Очень сложно заставить систему удалить данные или сделать что-то неотменяемое.

В Git файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. «Зафиксированный» значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующую фиксацию.

4 Работа с репозиторием с использованием GitHub и Visual Studio

Сервис GitHub – это один из крупнейших веб-сервисов для хостинга IT-проектов и их совместной разработки. Сервис основан на системе контроля версий Git.

Для работы с сервисом необходимо зарегистрироваться. После создания репозитория будет доступна возможность его клонирования (физического переноса файлов репозитория на локальную машину для дальнейшей работы или переноса репозитория на другой сервис контроля версий, поддерживающий Git). Клонирование возможно по двум протоколам HTTPS и SSH.

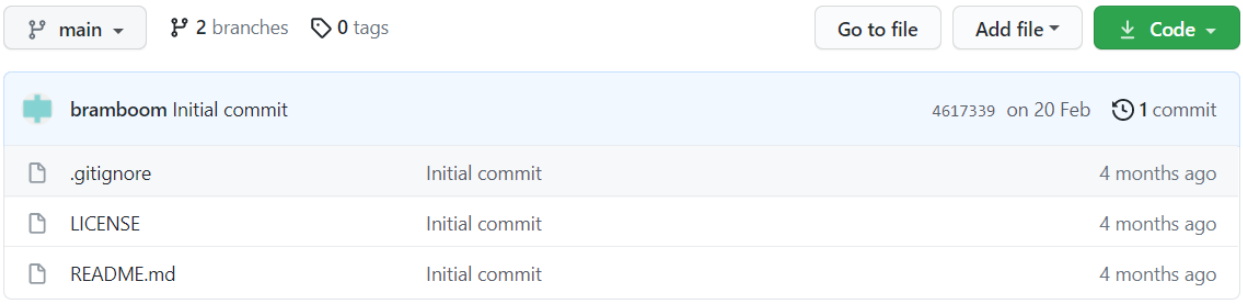

Рисунок 4.1 – Создание репозитория и главной ветви main

Произведено клонирование репозитория на локальный компьютер в Visual Studio. На рисунке 4.2 показан результат клонирования.

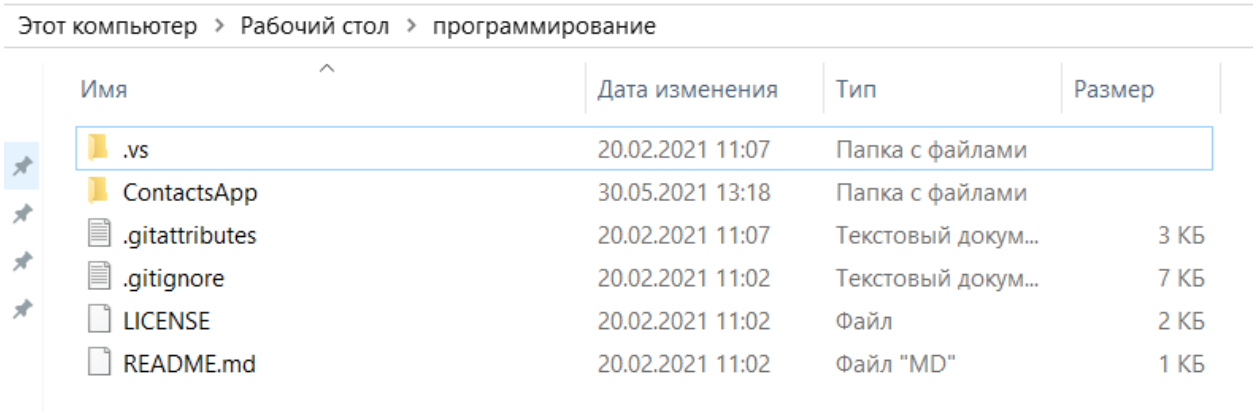

Рисунок 4.2 – Скриншот папки клонированного репозитория

По заданию была создана новая ветвь – develop, в которой было создано новое решение с исполняемым проектом Windows Forms (проект пользовательского интерфейса). После чего, эта ветвь была сохранена в репозитории с помощью синхронизации. Структура файлов репозитория в ветке develop стала выглядеть так (рисунок 4.3):

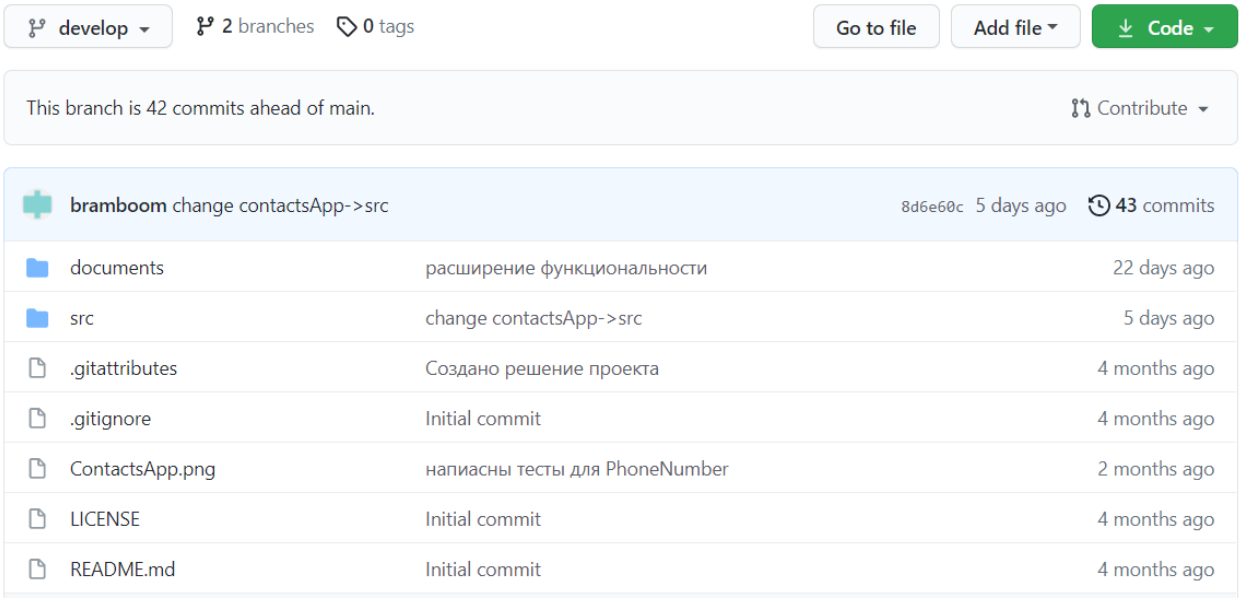

Рисунок 4.3 – Структура файлов репозитория в ветке develop

После чего в решение добавлена библиотека классов, которая иначе называется проект логики. В проект пользовательского интерфейса добавлена ссылка на библиотеку классов, которая впоследствии была подключена в коде пользовательского интерфейса. После завершения этой процедуры была сделана фиксация результатов и история ветки develop стала следующей (рисунок 4.4):

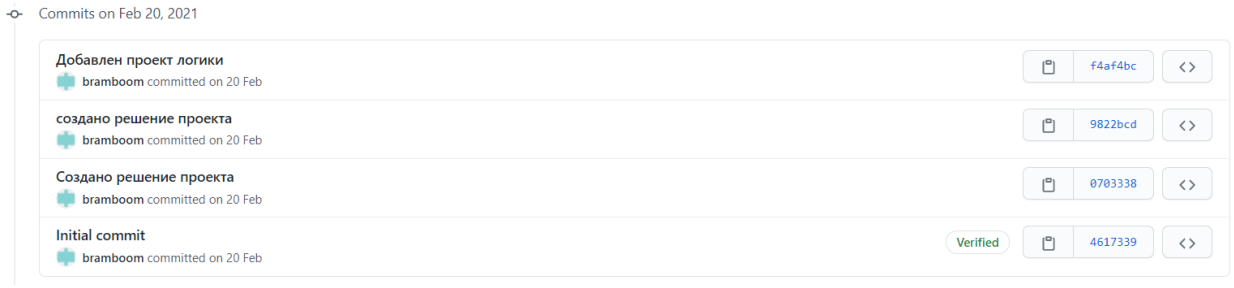

Рисунок 4.4 – История фиксаций ветки develop

5 Заключение

В ходе лабораторной работы изучен пакет программ, используемых при разработке декстоп-приложений и получены умения их развертки на рабочей машине