Министерство науки и высшего образования Российской Федерации

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Документация к проекту

Планировщик полетов

По дисциплине Технологии программирования

Выполнили:

Студенты группы 23-ПМ-1,2

Корноухов Илья

Смородин Дмитрий

Кабанов Дамир

Субботин Николай

Нижний Новгород 2024

Содержание

[Список требований 3](#_Задача:)

[Описание и идея проекта 4](#_Теоретическая_справка)

[Архитектура приложения 10](#_Реализация)

[Тестирование приложения 12](#_Результат_работы_программы)

Список требований

Цели:

1. Научится работать в команде
2. Научится пользоваться Git
3. Разработать документацию
4. Реализовать сервер и клиент
5. Провести тестирование программы

Задачи:

1. Придумать идею
2. Разработать архитектуру программы
3. Распределить задачи по написанию программы
4. Провести код review
5. Разработать документацию и презентацию для проекта
6. Сделать интерфейс для удобного использования

Описание и идея проекта

Название проекта – Планировщик полетов.

Идея проекта – Реализация клиент-серверного приложения для управления авиарейсами, где сервер отвечает за обработку данных, а клиент предоставляет удобный интерфейс для работы пользователя. Программа разработана на языках программирования Java и C++.

Возможности пользователя:

1. Просмотр аэропортов
2. Создание и удаление аэропортов
3. Обновление аэропортов
4. Создание и удаление самолетов
5. Изменение самолетов
6. Создание и просмотр полетов
7. Просмотр диспетчеров
8. Проматывание времени

Реализация программы:

Для написания программы клиенты использовался C#. Были использованы такие элементы как:

1. Label – для подсказок и вывода необходимой для пользователя информации
2. Button – для перехода программы в разные состояния (например от состояния меню с выбором действий к состоянию ввода пациента)
3. TextBox – для ввода данных пациентов
4. TrackBar – для удобного выбора количество дней для перемотки

Программа клиента собрана в exe файл, т.е. для работы пользователю нужно запустить один файл.

Когда пользователь нажимает кнопки, клиент отправляет необходимую информацию серверу о переходе из одного состояния к другому.

Состояния программы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Состояние** | **Пользователь** | **Клиент** | **Сервер** | **Полученное состояние** |
| Стартовое | Нажатие кнопки «START» | Отправляет информацию о том, что пользователь запустил программу | Получает информацию о том, что пользователь запустил программу | Выбор основного действия |
| Нажатие кнопки «ВЫХОД» | - | Закрывает соединение с клиентом | Закрытие программы |
| Выбор основного действия | Нажатие кнопки «Работа с пациентами» | Отправляет серверу «1» | Оператор switch получает «1» и переходит ко второму оператору switch | Работа с пациентами |
| Нажатие кнопки «Промотать время» | Отправляет серверу «2» | Оператор switch получает «2» и ждет сообщение от клиента | Работа со временем |
| Нажатие кнопки «ВЫХОД» | - | Закрывает соединение с клиентом | Закрытие программы |
| Работа с пациентами | Нажатие кнопки «Создать список пациентов» | Отправляет серверу «1» | Оператор switch получает «1»,  Запускает функцию create\_patients и ждет сообщение от клиента | Ввод количества пациентов |
| Нажатие кнопки «Добавить пациента» | Отправляет серверу «2» | Оператор switch получает «2»,  Запускает функцию add\_patients и ждет сообщение от клиента | Ввод пациента |
| Нажатие кнопки «Поиск пациента» | Отправляет серверу «3» | Оператор switch получает «3»,  Запускает функцию search\_patients и ждет сообщение от клиента | Поиск пациента |
| Нажатие кнопки «Удаление пациента» | Отправляет серверу «4» | Оператор switch получает «4»,  Запускает функцию delete\_patients и ждет сообщение от клиента | Удаление пациента |
| Нажатие кнопки «Показать список пациентов» | Отправляет серверу «5» | Оператор switch получает «5»,  Запускает функцию print\_patients и передает строки с данными о пациентах клиенту | Вывод пациентов |
| Нажатие кнопки «НАЗАД» | Отправляет серверу сообщение «BACK» | Получает сообщение «BACK» и возвращается на предыдущий switch | Выбор основного действия |
| Работа со временем | Выбор количества дней для перемотки и нажатие кнопки «Отправить» | Отправляет серверу введенное количество | Получает количество дней и передает их функции advance\_day, которая уменьшает параметр days для каждого пациента | Выбор основного действия |
| Нажатие кнопки «НАЗАД» | Отправляет серверу сообщение «BACK» | Получает сообщение «BACK» и отменяет перемотку времени | Выбор основного действия |
| Ввод количества пациентов | Ввод количества пациентов в текстовое поле и нажатие кнопки «Отправить» | Отправляет серверу введенное количество | Функция create\_patient получает количество дней и запускает цикл, запрашивая данные пациента указанное количество раз | Ввод пациента |
| Нажатие кнопки «НАЗАД» | Отправляет серверу сообщение «BACK» | Получает сообщение «BACK» и отменяет ввод клиентов | Работа с пациентами |
| Ввод пациента | Ввод данных пациента в форму и нажатие кнопки «Отправить» | Отправляет серверу строку с данными пациента | Функция create\_patient получает данные пациента, создает экземпляр класса Patient, записывает данные в него и добавляет экземпляр в общий список пациентов | Если оставшееся количество пациентов больше или равно 1, то очищает текстовые поля и остается здесь же, иначе переходит на «Работа с пациентами» |
| Поиск пациента | Ввод фамилии пациента в текстовое поле и нажатие кнопки «Отправить» | Отправляет серверу фамилию пациента | Функция search\_patient получает фамилия, проходится по списку клиентов, если находит такого пациента, то отправляет строку с его данными, иначе передает сообщение о том, что такого пациента нет | Результат поиска |
| Нажатие кнопки «НАЗАД» | Отправляет серверу сообщение «BACK» | Получает сообщение «BACK» и отменяет поиск клиента | Работа с пациентами |
| Результат поиска | Нажатие кнопки «НАЗАД» | Отправляет серверу сообщение «BACK» | Получает сообщение «BACK» и возвращается в цикл | Работа с пациентами |
| Удаление пациента | Ввод фамилии пациента в текстовое поле и нажатие кнопки «Отправить» | Отправляет серверу фамилию пациента | Функция delete\_patient получает фамилию, проходится по списку клиентов, если находит такого пациента, то отправляет сообщение об успешном удалении, иначе отправляет сообщение о том, что такого пациента нету | Результат удаления |
| Нажатие кнопки «НАЗАД» | Отправляет серверу сообщение «BACK» | Получает сообщение «BACK» и отменяет удаление клиента | Работа с пациентами |
| Результат удаления | Нажатие кнопки «НАЗАД» | Отправляет серверу сообщение «BACK» | Получает сообщение «BACK» и возвращается в цикл | Работа с пациентами |
| Вывод пациентов | Нажатие кнопки «НАЗАД» | Отправляет серверу сообщение «BACK» | Получает сообщение «BACK» и возвращается в цикл | Работа с пациентами |

Архитектура приложения

Сервер представляет собой консольное приложение на C++. Программа собирается с файла Hospital Proj.cpp, где находится функция main. В векторе temp, который объявляется в начале кода, хранятся объекты класса Patient. Свойства и методы этого класса написаны в файле Patient.h.

База данных сервера представляет собой файл patient.txt, в котором хранится число – количество пациентов и данные о пациентах. С помощью функции read\_patients, реализованной в файле database.cpp, в вектор записываются объекты класса Patient. Для этого реализована перегрузка оператора ввода в файле Func.cpp.

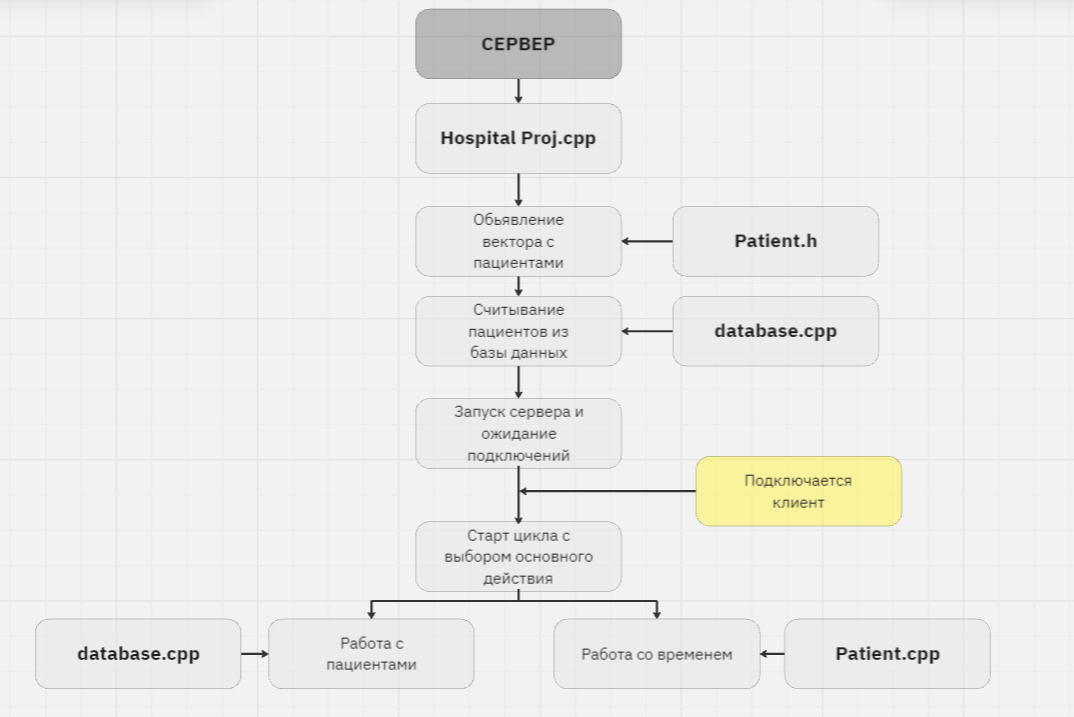
Когда данные о пациентах находятся непосредственно на сервере, производим инициализацию Winsock, библиотеки для работы с сокетами. Создаем сокет сервера, настраиваем адрес сервера, привязываем сокет к IP-адресу и порту. После этого сервер готов принимать подключение от клиента. 

Рис. 1 Архитектура сервера

При запуске программы клиента, на сервере принимается подключение и создается сокет клиента.

Запускается бесконечный цикл, который можно остановить, напрямую закрыв программу сервера. Это сделано для того, чтобы клиент не мог остановить программу сервера и мог открывать и закрывать приложение любое количество раз, пока сервер включен.

В цикле принимается сообщение от клиента, содержащее код действия. Этот код попадает в оператор switch, если он равен «1», то запускается новый бесконечный цикл с другим оператором switch, где оператор также принимает код и выполняет соответствующие функции для работы с пациентом. Если же код равен «2», то программа будет ждать сообщение от клиента с числом. Это число является количеством дней для перемотки. Программа передает это число в функцию advance\_day, реализованную в Patient.cpp.

Тестирование приложения

Для тестирования приложения составлен unit tests проверяющий работоспособность слой репозиториев в клиенте.

**Автоматический тест**

Для автоматического тестирования, напишем программу на C#, которая будет запускать приложение клиента, создавать список клиентов и запрашивать вывод этого списка.

Для реализации будем использовать библиотеку FlaUI для .NET.

Входные данные:

Ввод одного пациента, запрос вывода информации о пациентах

Выход:

