Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Петрозаводский государственный университет»

Физико-технический институт

Кафедра информационно-измерительных систем, электроники и автоматики

ПЕКАРНЯ

отчет по лабораторной работе

Работу выполнили:

Студенев В.A. группа 21316

Соловьев Н.В. группа 21316

Зинков С.П. группа 21316

Иванов Р.В. группа 21316

Научный руководитель:  
 канд. физ.-мат. наук, доцент  
 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Бульба   
 « » декабря 2023 г.

Петрозаводск  
2023 год

Оглавление

[**Цель** 4](#_Toc154736372)

[**Программная реализация** 4](#_Toc154736373)

[**Краткое словесное описание сюжета** 4](#_Toc154736374)

[**Описание имеющихся у заказчика материалов** 5](#_Toc154736375)

[**Актеры** 6](#_Toc154736376)

[**Диаграмма вариантов использования** 6](#_Toc154736377)

[**Описание каждого варианта использования** 7](#_Toc154736378)

[**Диаграммы действий** 8](#_Toc154736379)

[**Предварительный список существительных** 9](#_Toc154736380)

[**Уточненный список существительных** 10](#_Toc154736381)

[**Список атрибутов** 10](#_Toc154736382)

[**Список сообщений** 11](#_Toc154736383)

[**Диаграмма классов** 13](#_Toc154736384)

[**Диаграмма последовательностей** 14](#_Toc154736385)

[**Листинг программы** 16](#_Toc154736386)

[ingredient.h 16](#_Toc154736387)

[ingredientforrecepi.h 16](#_Toc154736388)

[listingredients.h 17](#_Toc154736389)

[listrecepi.h 17](#_Toc154736390)

[listreports.h 18](#_Toc154736391)

[listworkers.h 18](#_Toc154736392)

[product.h 19](#_Toc154736393)

[report.h 19](#_Toc154736394)

[resepi.h 20](#_Toc154736395)

[userinterface.h 21](#_Toc154736396)

[worker.h 21](#_Toc154736397)

[ingredient.cpp 22](#_Toc154736398)

[ingredientforrecepi.cpp 23](#_Toc154736399)

[listingredients.cpp 24](#_Toc154736400)

[listrecepi.cpp 27](#_Toc154736401)

[listreports.cpp 29](#_Toc154736402)

[listworkers.cpp 31](#_Toc154736403)

[product.cpp 33](#_Toc154736404)

[recepi.cpp 34](#_Toc154736405)

[report.cpp 38](#_Toc154736406)

[userinterface.cpp 41](#_Toc154736407)

[worker.cpp 44](#_Toc154736408)

[main.cpp 45](#_Toc154736409)

[**Руководство пользователя** 47](#_Toc154736410)

[**История проекта на GitHub** 54](#_Toc154736411)

[**Заключение:** 55](#_Toc154736412)

**Цель**

Рассмотреть простой пример выполнения заказа на разработку программного обеспечения и, в качестве самостоятельного задания, выполнить аналогичный проект

**Программная реализация**

Для реализации поставленной задачи была использована среда программирования Qt Creator 5.4.2, в качестве языка программирования был выбран С++. В процессе работы над проектом созданы заголовочные файлы:

1. ingredient.h
2. ingredientforrecepi.h
3. resepi.h
4. listingredients.h
5. listproducts.h
6. listrecepi.h
7. listreports.h
8. listworkers.h
9. product.h
10. userinterface.h
11. worker.h

**Краткое словесное описание сюжета**

Для небольшой пекарни требуется программа для ведения учета сырья и готовой продукции. В пекарне имеется 3 должности: администратор, пекарь и заведующий складом.

Пекарь осуществляет производство хлебобулочных изделий по заданным рецептам из ингредиентов, хранящихся на складе. В программе он должен осуществлять добавление новой продукции, также иметь возможность просмотра имеющихся рецептов, количества ингредиентов на складе и количества всей готовой продукции, произведенной за текущий день.

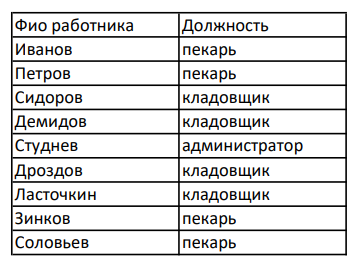
Заведующий складом занимается обеспечением склада, следит за пополнением сырья.

Администратор главный человек на производстве, который контролирует все процессы, осуществляет добавление новых рецептов, новых видов сырья и новых пользователей в системе, также может взять на себя обязанности других сотрудников. Администратор должен иметь возможность просмотра всей готовой продукции за любой день, когда пекарня работала.

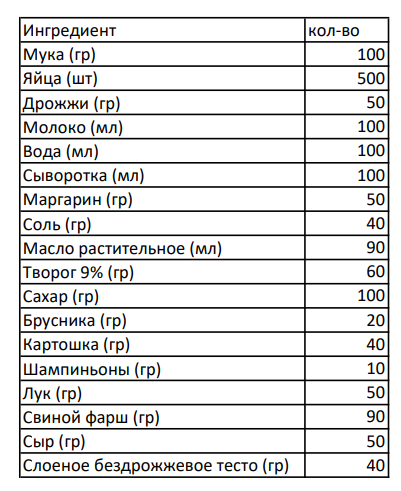
Каждый сотрудник должен заходить в систему под своим уникальным именем и паролем. При добавлении новой готовой продукции система должна автоматически списывать нужное количество ингредиентов согласно рецепту.

**Описание имеющихся у заказчика материалов**

* Списки сотрудников пекарни с их должностью



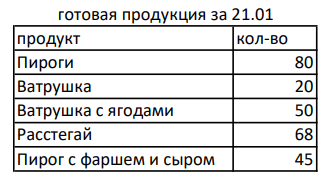
* Учетная книга склада



* Список рецептов

1. Ватрушка: 50 шт.
   * Мука – 400 гр.
   * Вода – 400 мл
   * Яйца – 5 шт.
   * Дрожжи – 50 гр.
   * Соль – 25 гр.
   * Творог 9% - 500 гр.
   * Сахар – 300 гр.
2. Пироги: (на 30 шт.)
   * Мука – 400 гр.
   * Сыворотка – 400 гр.
   * Дрожжи – 30 гр.

* Книга произведенной продукции



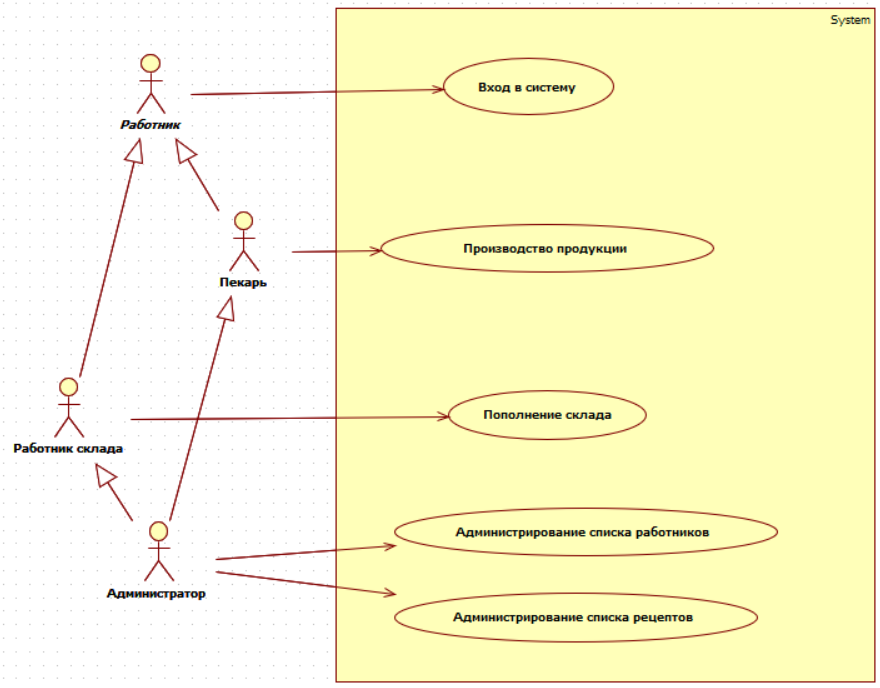
**Актеры**

Администратор – контролирует все процессы, осуществляет добавление новых рецептов, новых видов сырья и новых пользователей в системе, также может взять на себя обязанности других сотрудников

Работник склада - занимается обеспечением склада, следит за пополнением сырья.

Пекарь – может добавлять новую продукцию на склад, имеет возможность просмотра имеющихся рецептов, количества ингредиентов на складе и количества всей готовой продукции, произведенной за текущий день

**Диаграмма вариантов использования**

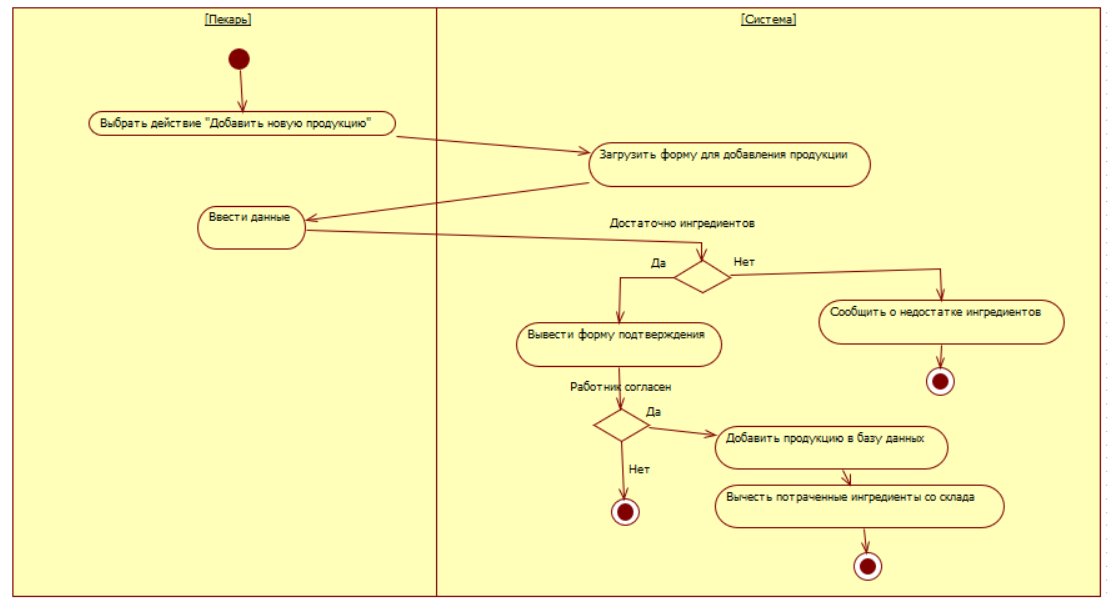


**Описание каждого варианта использования**

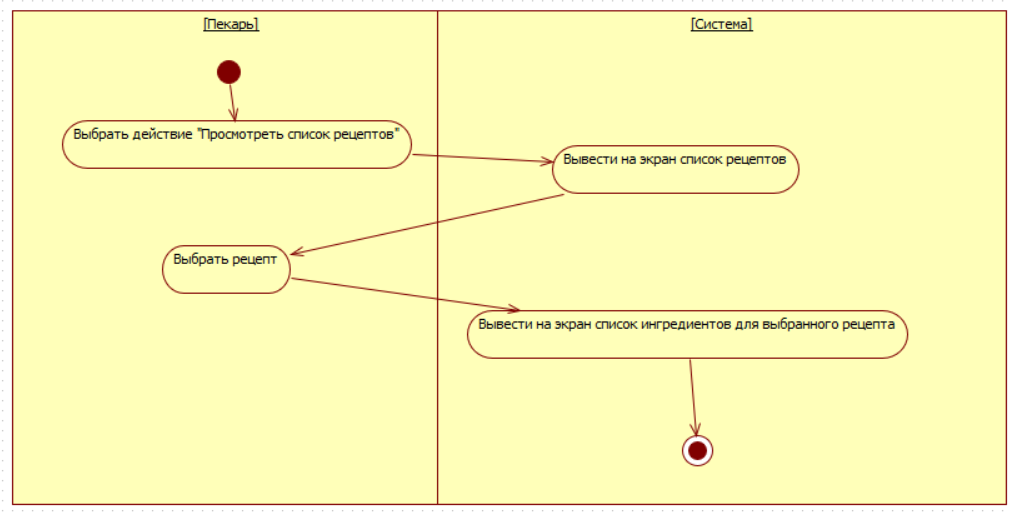
1. Вход в систему - описывает вход работника в систему учета производства пекарни. От работника требуется ввести его уникальный логин и пароль, которые он получит от администратора.
2. Производство продукции - пекарь производит продукцию из ингредиентов, которые имеются на складе. Потом он добавляет произведенную продукцию в отчет за текущий день.
3. Пополнения склада - когда поставщики привозят ингредиенты в пекарню, работник склада добавляет ингредиенты в базу данных склада.
4. Администрирование списка работников - у администратора есть доступ к списку работников, где хранятся логины и пароли всех работников. Он может просмотреть список работников, а также добавить новых работников и удалить уже имеющиеся.
5. Администрирование списка рецептов - у администратора есть доступ к списку рецептов, где хранятся названия и состав каждого рецепта. Он может просмотреть список рецептов, а также добавить новые и удалить уже имеющиеся.

**Диаграммы действий**

Добавление готовой продукции:



Просмотр рецептов:



**Предварительный список существительных**

* + Работник
  + Пекарь
  + Работник склада
  + Админ
  + Ингредиент
  + Рецепт
  + Список ингредиентов
  + Список рецептов
  + Список работников
  + Склад
  + Стартовый экран
  + Меню
  + Список отчетов
  + Отчет
  + Готовая продукция
  + Экран добавления готовой продукции
  + Экран ввода логина и пароля
  + Экран главного меню
  + Экран списка рецептов
  + Экран списка ингредиентов
  + Экран списка работников
  + Экран добавления продукции
  + Экран добавления рецептов
  + Экран добавления ингредиентов
  + Экран добавления работников
  + Отчет
  + Список отчетов
  + Контейнер

**Уточненный список существительных**

1. Работник – запись о пользователе системы. Включает в себя логин и пароль для входа в систему, а также права доступа. Существует три типа прав доступа: администратор, пекарь и работник склада. В зависимости от прав доступа определяются возможности пользователя в системе.
2. Список работников – список всех пользователей системы. Только администратор управляет списком рабочих. При этом он не может создать или удалить работника с правами доступа администратора. В системе всегда существует ровно один администратор.
3. Ингредиент – запись о исходной продукции. Включает в себя наименование, количество и единицы измерения. Используется для описания содержимого склада и описания рецепта.
4. Рецепт – список ингредиентов. Включает в себя нужные ингредиенты для производства одной единицы готовой продукции. Ингредиенты в рецепт добавляются на основе содержимого склада.
5. Список рецептов – список всех имеющихся в системе рецептов.
6. Склад - список ингредиентов. Включает в себя в себя доступные для производства ингредиенты. При производстве продукции со склада изымается нужное количество ингредиентов.
7. Продукция – запись о готовой продукции. Включает в себя наименование и количество. Добавляется в систему согласно списку рецептов и строго в текущий день.
8. Отчет по готовой продукции – список продукции. Включает в себя всю произведённую продукцию за определённую дату. Добавление новой осуществляется только в текущий день.
9. Список отчётов – список отчётов за все дни, когда система была активна. Новые отчёты добавляются в систему автоматически, при первом входе в систему за день.
10. Главный экран – осуществляет доступ к нужным элементам системы согласно правам доступа.
11. Экран входа – осуществляет проверку при входе в систему на основе введённого логина и пароля. При успешной проверке определяет права доступа и предоставляет нужную конфигурацию системы.

**Список атрибутов**

1. Работник
   1. Логин – уникальный идентификатор в системе
   2. Пароль
   3. Права доступа– 3 строго определённых варианта
2. Список работников
   1. Имя
   2. Список работников
3. Ингредиент
   1. Наименование - уникальный идентификатор в отдельном списке
   2. Количество – положительное целое число
   3. Единицы измерения - 3 строго определённых варианта, является дополнительной информацией
4. Рецепт
   1. Имя - уникальный идентификатор в системе
   2. Список ингредиентов
   3. Текущий ингредиент – указатель на элемент в списке с которым можно взаимодействовать в данный момент
5. Список рецептов
   1. Имя - уникальный идентификатор в системе
   2. Список рецептов
   3. Текущий рецепт – указатель на элемент в списке с которым можно взаимодействовать в данный момент
6. Склад
   1. Имя - уникальный идентификатор в системе
   2. Список ингредиентов
   3. Текущий ингредиент – указатель на элемент в списке с которым можно взаимодействовать в данный момент
7. Продукция
   1. Наименование - уникальный идентификатор в отдельном списке
   2. Количество – положительное целое число
   3. Единицы измерения - 3 строго определённых варианта, является дополнительной информацией
8. Отчет по готовой продукции
   1. Имя - уникальный идентификатор в системе
   2. Список продукции
9. Список отчётов
   1. Дата - уникальный идентификатор в системе
   2. Список отчётов
10. Главный экран
11. Экран входа

**Список сообщений**

1. Войти в систему

Пользователь вводит логин и пароль, эти данные запрашиваются из загруженного списка и проверяются там, если данные сходятся, пользователь заходит в систему, если нет то выводится сообщение об ошибке. Затем проверяются права доступа пользователя, в зависимости от которых определяются его возможности в системе. Для каждой должности существуют свои открытые и недоступные функции.

1. Открыть главный экран

В зависимости от прав доступа перед пользователем отображаются доступные элементы системы с которыми пользователь может взаимодействовать.

1. Администратор
   1. Список работников
   2. Склад
   3. Список рецептов
   4. Список отчётов
2. Пекарь
   1. Склад
   2. Список рецептов
   3. Текущий отчёт
3. Работник склада
   1. Склад
4. Открыть список

Запрос на просмотр необходимого списка для его последующего использования пользователем.

1. Добавить элемент в список

В открытом списке пользователь добавляет новый элемент. Возможность добавления зависит от прав доступа.

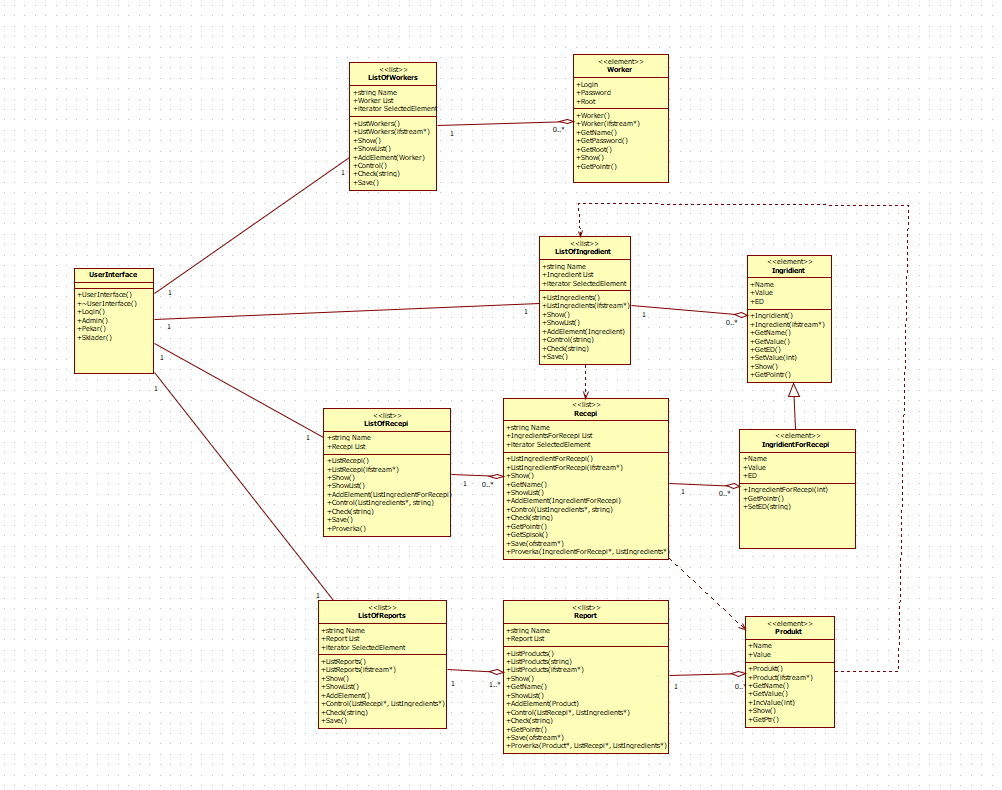
1. Администратор
   1. Список работников
   2. Склад
   3. Список рецептов
   4. Текущий отчёт
2. Пекарь
   1. Текущий отчёт
3. Работник склада
   1. Склад

Добавление ингредиента в рецепт возможно только при наличии его на складе. Добавление продукции возможно только при наличии нужного рецепта и достаточного количества ингредиентов на складе, а затем со склада списываются затраченные инциденты.

1. Удалить элемент из списка

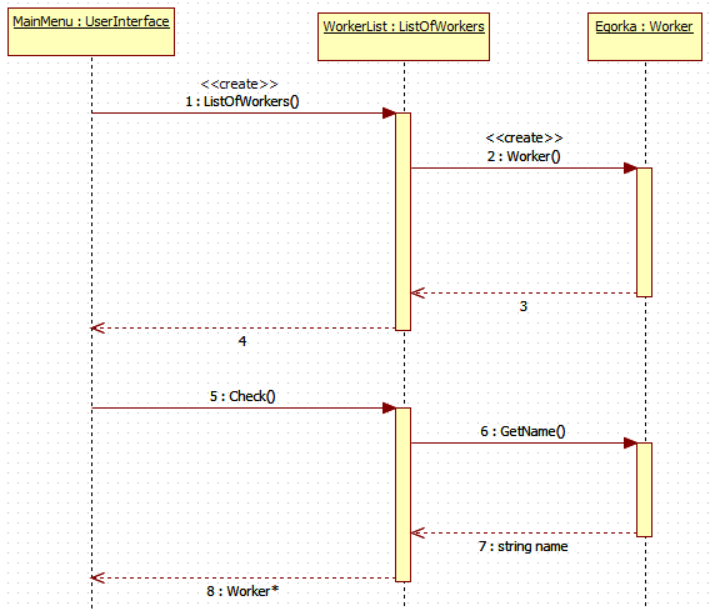
Таким правом обладает только администратор. Он может удалить любые элементы списка, исключая отчет, запись о администраторе и готовую продукцию

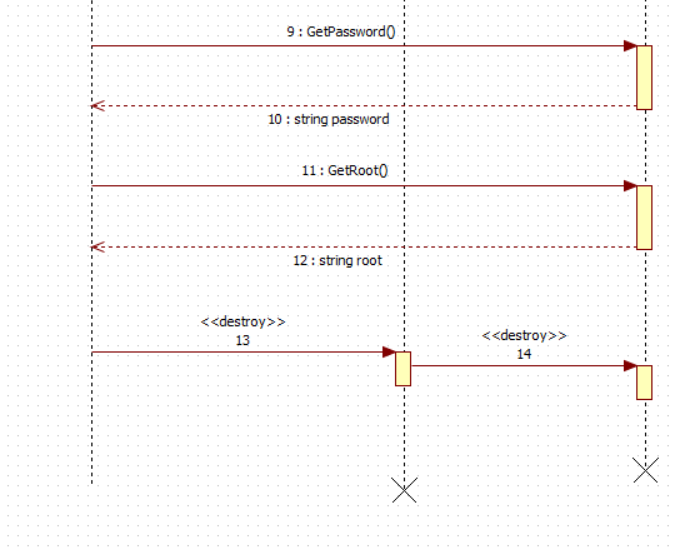
**Диаграмма классов**



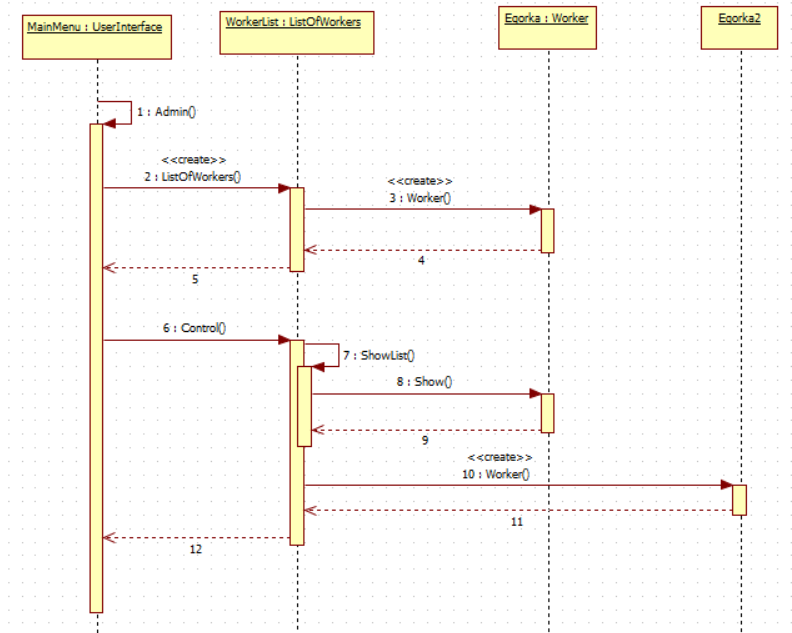
**Диаграмма последовательностей**

Вход в систему

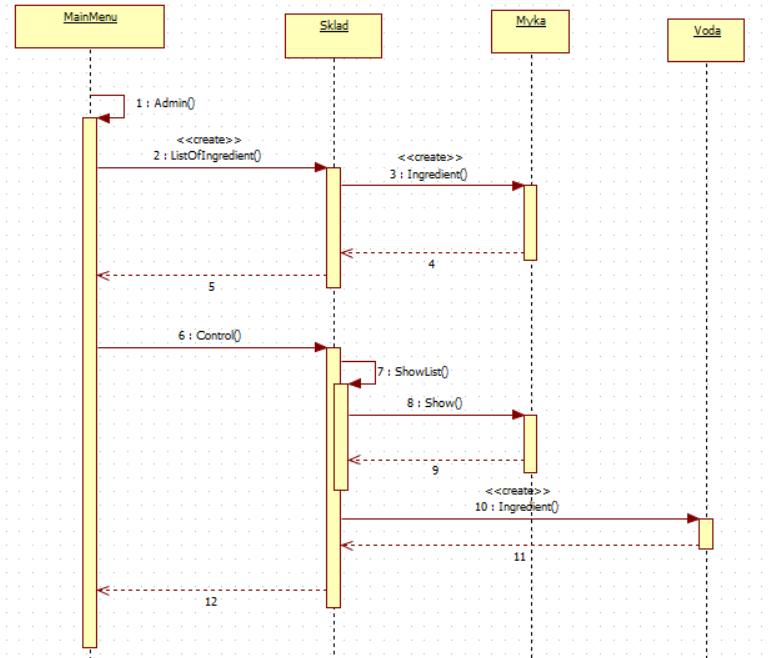




Добавление работника



Добавление ингредиента



**Листинг программы**

## ingredient.h

#ifndef INGREDIENT\_H

#define INGREDIENT\_H

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

class Ingredient {

protected:

string Name;

int Value;

string ED;

public:

Ingredient();

Ingredient(int);

Ingredient(ifstream\*);

string GetName();

int GetValue();

string GetED();

void SetValue(int);

void shou();

Ingredient\* GetPointr();

};

#endif // INGREDIENT\_H

## ingredientforrecepi.h

#ifndef INGREDIENTFORRECEPI\_H

#define INGREDIENTFORRECEPI\_H

#include <string>

#include <iostream>

#include <fstream>

#include "ingredient.h"

using namespace std;

class IngredientForRecepi : public Ingredient{

public:

IngredientForRecepi();

IngredientForRecepi(int);

IngredientForRecepi(ifstream\*);

IngredientForRecepi\* GetPointr();

void SetED(string);

};

#endif // INGREDIENTFORRECEPI\_H

## listingredients.h

#ifndef LISTINGREDIENTS\_H

#define LISTINGREDIENTS\_H

#include <iostream>

#include <list>

#include <string>

#include <conio.h>

#include <cstdlib>

#include <windows.h>

#include <fstream>

#include <ctime>

#include "ingredient.h"

using namespace std;

class ListIngredients {

private:

list<Ingredient> Spisok;

list<Ingredient>::iterator SelectedElement;

string Name;

public:

ListIngredients();

ListIngredients(ifstream\*);

void shou();

void shoulist();

void AddElement(Ingredient);

void Control(string);

Ingredient\* Check(string);

void Save();

};

#endif // LISTINGREDIENTS\_H

## listrecepi.h

#ifndef LISTRECEPI\_H

#define LISTRECEPI\_H

#include <iostream>

#include <list>

#include <string>

#include <conio.h>

#include <cstdlib>

#include <windows.h>

#include <fstream>

#include <ctime>

#include "resepi.h"

#include "listingredients.h"

using namespace std;

class ListRecepi {

private:

list<ListIngredientForRecepi> Spisok;

list<ListIngredientForRecepi>::iterator SelectedElement;

string Name;

public:

ListRecepi();

ListRecepi(ifstream\*);

void shou();

void shoulist();

void AddElement(ListIngredientForRecepi);

void Control(ListIngredients\*, string);

ListIngredientForRecepi\* Check(string);

void Save();

bool Proverka();

};

#endif // LISTRECEPI\_H

## listreports.h

#ifndef LISTREPORTS\_H

#define LISTREPORTS\_H

#include <iostream>

#include <list>

#include <string>

#include <conio.h>

#include <cstdlib>

#include <windows.h>

#include <fstream>

#include <ctime>

#include "listproducts.h"

using namespace std;

class ListReports {

private:

list<ListProducts> Spisok;

list<ListProducts>::iterator SelectedElement;

string Name;

public:

ListReports();

ListReports(ifstream\*);

void shou();

void shoulist();

void AddElement(ListProducts);

void Control(ListRecepi\*,ListIngredients\*);

ListProducts\* Check(string);

void Save();

};

#endif // LISTREPORTS\_H

## listworkers.h

#ifndef LISTREPORTS\_H

#define LISTREPORTS\_H

#include <iostream>

#include <list>

#include <string>

#include <conio.h>

#include <cstdlib>

#include <windows.h>

#include <fstream>

#include <ctime>

#include "listproducts.h"

using namespace std;

class ListReports {

private:

list<ListProducts> Spisok;

list<ListProducts>::iterator SelectedElement;

string Name;

public:

ListReports();

ListReports(ifstream\*);

void shou();

void shoulist();

void AddElement(ListProducts);

void Control(ListRecepi\*,ListIngredients\*);

ListProducts\* Check(string);

void Save();

};

#endif // LISTREPORTS\_H

## product.h

#ifndef PRODUCT\_H

#define PRODUCT\_H

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

class Product {

private:

string Name;

int Value;

public:

Product();

Product(ifstream\*);

string GetName();

int GetValue();

void IncValue(int);

void shou();

Product\* GetPointr();

};

#endif // PRODUCT\_H

## report.h

#ifndef LISTPRODUCTS\_H

#define LISTPRODUCTS\_H

#include <iostream>

#include <list>

#include <string>

#include <conio.h>

#include <cstdlib>

#include <windows.h>

#include <fstream>

#include <ctime>

#include "product.h"

#include "listingredients.h"

#include "ingredient.h"

#include "listrecepi.h"

#include "resepi.h"

#include "ingredientforrecepi.h"

using namespace std;

class ListProducts {

private:

list<Product> Spisok;

list<Product>::iterator SelectedElement;

string Name;

public:

ListProducts();

ListProducts(string);

ListProducts(ifstream\*);

void shou();

string GetName();

void shoulist();

void AddElement(Product);

void Control(ListRecepi\*,ListIngredients\*);

Product\* Check(string);

ListProducts\* GetPointr();

void Save(ofstream\*);

bool Proverka(Product\*,ListRecepi\*,ListIngredients\*);// реализовать

};

#endif // LISTPRODUCTS\_H

## resepi.h

#ifndef LISTINGREDIENTFORRECEPI\_H

#define LISTINGREDIENTFORRECEPI\_H

#include <iostream>

#include <list>

#include <string>

#include <conio.h>

#include <cstdlib>

#include <windows.h>

#include <fstream>

#include <ctime>

#include "ingredientforrecepi.h"

#include "listingredients.h"

using namespace std;

class ListIngredientForRecepi {

private:

list<IngredientForRecepi> Spisok;

list<IngredientForRecepi>::iterator SelectedElement;

string Name;

public:

ListIngredientForRecepi();

ListIngredientForRecepi(ifstream\*);

void shou();

string GetName();

void shoulist();

void AddElement(IngredientForRecepi);

void Control(ListIngredients\*, string);

IngredientForRecepi\* Check(string);

ListIngredientForRecepi\* GetPointr();

list<IngredientForRecepi> GetSpisok();

void Save(ofstream\*);

bool Proverka(IngredientForRecepi\*,ListIngredients\*);

};

#endif // LISTINGREDIENTFORRECEPI\_H

## userinterface.h

#ifndef USERINTERFACE\_H

#define USERINTERFACE\_H

#include <fstream>

#include <string>

#include "listworkers.h"

#include "listingredients.h"

#include "listrecepi.h"

#include "listreports.h"

using namespace std;

class UserInterface

{

public:

UserInterface();

void Login();

void Admin();

void Pekar();

void Sklader();

};

#endif // USERINTERFACE\_H

## worker.h

#ifndef WORKER\_H

#define WORKER\_H

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

class Worker {

private:

string Login;

string Password;

string Root;

public:

Worker();

Worker(ifstream\*);

string GetName();

string GetPassword();

string GetRoot();

void shou();

Worker\* GetPointr();

};

#endif // WORKER\_H

## ingredient.cpp

#include "ingredient.h"

Ingredient::Ingredient()

{

string n, e, temp\_v;

char mas[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

unsigned int count = 0;

cout << "Vvedite naimenovanie: " << endl;

cin >> n;

\_flushall();

Name = n;

cout << "Vvedite kol-vo: " << endl;

cin >> temp\_v;

\_flushall();

for(unsigned int i = 0; i < temp\_v.size(); ++i){

bool exists = false;

for (int j: mas){

if (j == temp\_v[i]) {

exists = true;

break;

}

}

if(exists){

count++;

}

}

if(count == temp\_v.size()){

Value = stoi(temp\_v);

cout << "Neobhodimo ukazat edinicu izmereniya ingredienta." << endl;

cout << "Tri edinicy izmereniya: 1 - gr. 2 - sht. 3 - ml. (ukazhite cifru)" << endl;

cout << "Edinica izmereniya ingredienta: ";

cin >> e;

\_flushall();

if(e == "1"){

ED = "gr.";

}

else if(e == "2"){

ED = "sht.";

}

else if(e == "3"){

ED = "ml.";

}

else{

ED = "";

cout << "Vy vveli nekorrektnoe chislo!!!" << endl;

system("pause");

}

}

else{

temp\_v = "-1";

Value = stoi(temp\_v);

ED = "";

cout << "Vy vveli nekorrektnoe chislo!!!" << endl;

system("pause");

}

}

Ingredient::Ingredient(int a)

{

}

Ingredient::Ingredient(ifstream \*file)

{

string N;

string temp\_V;

string ED;

getline(\*file, N, ' ');

getline(\*file, temp\_V, ' ');

getline(\*file, ED);

Name = N;

Value = stoi(temp\_V);

this->ED = ED;

}

string Ingredient::GetName()

{

return Name;

}

int Ingredient::GetValue()

{

return Value;

}

string Ingredient::GetED(){

return ED;

}

void Ingredient::SetValue(int Value){

this->Value = Value;

}

void Ingredient::shou()

{

cout << Name << " " << Value << " " << ED << " " ;

}

Ingredient\* Ingredient::GetPointr(){

return this;

}

## ingredientforrecepi.cpp

#include "ingredientforrecepi.h"

IngredientForRecepi::IngredientForRecepi(int a):Ingredient(a)

{

string n, e, temp\_v;

char mas[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

unsigned int count = 0;

cout << "Vvedite naimenovanie: " << endl;

cin >> n;

\_flushall();

Name = n;

cout << "Vvedite kol-vo: " << endl;

cin >> temp\_v;

\_flushall();

for(unsigned int i = 0; i < temp\_v.size(); ++i){

bool exists = false;

for (int j: mas){

if (j == temp\_v[i]) {

exists = true;

break;

}

}

if(exists){

count++;

}

}

if(count == temp\_v.size()){

Value = stoi(temp\_v);

}else{

cout << "Vy vveli nekorrektnoe chislo!!!" << endl;

system("pause");

Value = -1;

}}

IngredientForRecepi::IngredientForRecepi():Ingredient()

{

}

IngredientForRecepi::IngredientForRecepi(ifstream \*file):Ingredient(file)

{

}

IngredientForRecepi\* IngredientForRecepi::GetPointr()

{

return this;

}

void IngredientForRecepi::SetED(string ED){

this->ED = ED;

}

## listingredients.cpp

#include "listingredients.h"

ListIngredients::ListIngredients(){

}

ListIngredients::ListIngredients(ifstream \*file)

{

string N;

string last;

getline(\*file, N, ' ');

Name = N;

getline(\*file, last);

int count = 1;

while ((file->eof() != true) && (count <= stoi(last)))

{

Ingredient\* NewListIngredients = new Ingredient(file);

ListIngredients::AddElement(\*NewListIngredients);

delete NewListIngredients;

count++;

}

}

void ListIngredients::shou(){

cout << Name;

}

void ListIngredients::shoulist(){

cout << endl;

list<Ingredient>::iterator ItrForShouing;

ItrForShouing = Spisok.begin();

for (ItrForShouing = Spisok.begin(); ItrForShouing != Spisok.end(); ++ItrForShouing){

ItrForShouing->shou();

if (ItrForShouing == SelectedElement){

cout << " <---";

}

cout << endl;

}

}

void ListIngredients::AddElement(Ingredient x){

Spisok.push\_back(x);

}

void ListIngredients::Control(string root){

SelectedElement = Spisok.begin();

bool conec = true;

while (conec) {

system("cls");

if(root != "pekar"){

cout << "a - add ingredient " << endl;

}

if(root == "admin"){

cout << "d - delete ingredient " << endl;

}

ListIngredients::shou();

ListIngredients::shoulist();

switch (\_getch()){

case 80: //V

if (SelectedElement != Spisok.end())

{

SelectedElement++;

}

if (SelectedElement == Spisok.end())

{

SelectedElement--;

}

break;

case 72: // ^

if (SelectedElement != Spisok.begin())

{

SelectedElement--;

}

break;

case 'd':

if(!Spisok.empty() && root == "admin"){

Spisok.erase(SelectedElement);

SelectedElement = Spisok.begin();

}

break;

case 'a':{if(root != "pekar"){

Ingredient\* NewListIngredients = new Ingredient;

Ingredient\* TestListIngredients = ListIngredients::Check(NewListIngredients->GetName());

if(NewListIngredients->GetValue() != -1 && NewListIngredients->GetED() != ""){

if (TestListIngredients == nullptr){

ListIngredients::AddElement(\*NewListIngredients);

}

else if(TestListIngredients != nullptr && NewListIngredients->GetED() == TestListIngredients->GetED()){

TestListIngredients->SetValue(TestListIngredients->GetValue() + NewListIngredients->GetValue());

}

else{

cout << "neverno vvedeny dannye";

\_getch();

}

}

delete NewListIngredients;}

break;}

case 27: // esc

conec = false;

break;

default:{}

}

}

}

Ingredient\* ListIngredients::Check(string N){

list<Ingredient>::iterator ItrToFind;

ItrToFind = Spisok.begin();

for (ItrToFind = Spisok.begin(); ItrToFind != Spisok.end(); ++ItrToFind)

if (ItrToFind->GetName() == N){

return ItrToFind->GetPointr();

};

return nullptr;

}

void ListIngredients::Save(){

ofstream file\_w("D:/WorkFiles/sklad.txt", ios\_base::out | ios\_base::trunc);

file\_w << Name + ' ';

if(Spisok.size() == 0){

file\_w << Spisok.size();

}

else{

file\_w << Spisok.size() << endl;

}

for(SelectedElement = Spisok.begin(); SelectedElement != Spisok.end(); ++SelectedElement){

if(SelectedElement == --Spisok.end()){

file\_w << (\*SelectedElement).GetName() + ' ';

file\_w << to\_string((\*SelectedElement).GetValue()) + ' ';

file\_w << (\*SelectedElement).GetED();

}

else{

file\_w << (\*SelectedElement).GetName() + ' ';

file\_w << to\_string((\*SelectedElement).GetValue()) + ' ';

file\_w << (\*SelectedElement).GetED() << endl;

}

}

file\_w.close();

}

## listrecepi.cpp

#include "listrecepi.h"

ListRecepi::ListRecepi(){

}

ListRecepi::ListRecepi(ifstream \*file)

{

string N;

string last;

getline(\*file, N, ' ');

Name = N;

getline(\*file, last);

int count = 1;

while ((file->eof() != true) && (count <= stoi(last)))

{

ListIngredientForRecepi\* NewListRecepi = new ListIngredientForRecepi(file);

ListRecepi::AddElement(\*NewListRecepi);

delete NewListRecepi;

count++;

}

}

void ListRecepi::shou(){

cout << Name;

}

void ListRecepi::shoulist(){

cout << endl;

list<ListIngredientForRecepi>::iterator ItrForShouing;

ItrForShouing = Spisok.begin();

for (ItrForShouing = Spisok.begin(); ItrForShouing != Spisok.end(); ++ItrForShouing){

ItrForShouing->shou();

if (ItrForShouing == SelectedElement){

cout << " <---";

}

cout << endl;

}

}

void ListRecepi::AddElement(ListIngredientForRecepi x){

Spisok.push\_back(x);

}

void ListRecepi::Control(ListIngredients \*sklad, string root){

SelectedElement = Spisok.begin();

bool conec = true;

while (conec) {

system("cls");

if(root == "admin"){

cout << "d - delete recepi " << endl;

cout << "a - add recepi " << endl;

}

ListRecepi::shou();

ListRecepi::shoulist();

switch (\_getch()){

case 80: //V

if (SelectedElement != Spisok.end())

{

SelectedElement++;

}

if (SelectedElement == Spisok.end())

{

SelectedElement--;

}

break;

case 72: // ^

if (SelectedElement != Spisok.begin())

{

SelectedElement--;

}

break;

case 'd':

if(!Spisok.empty() && root == "admin"){

Spisok.erase(SelectedElement);

SelectedElement = Spisok.begin();

}

break;

case 'a':{if(root == "admin"){

ListIngredientForRecepi\* NewListRecepi = new ListIngredientForRecepi;

ListIngredientForRecepi\* TestListRecepi = ListRecepi::Check(NewListRecepi->GetName());

if (TestListRecepi == nullptr){

ListRecepi::AddElement(\*NewListRecepi);

}

delete NewListRecepi;}

break;}

case 13:

(\*SelectedElement).Control(sklad, root);

break;

case 27: // esc

conec = false;

break;

default:{}

}

}

}

ListIngredientForRecepi\* ListRecepi::Check(string N){

list<ListIngredientForRecepi>::iterator ItrToFind;

ItrToFind = Spisok.begin();

for (ItrToFind = Spisok.begin(); ItrToFind != Spisok.end(); ++ItrToFind)

if (ItrToFind->GetName() == N){

return ItrToFind->GetPointr();

};

return nullptr;

}

void ListRecepi::Save(){

ofstream file\_w("D:/WorkFiles/listofrecepi.txt", ios\_base::out | ios\_base::trunc);

file\_w << Name + ' ';

if(Spisok.size() == 0){

file\_w << Spisok.size();

}

else{

file\_w << Spisok.size() << endl;

}

for(SelectedElement = Spisok.begin(); SelectedElement != Spisok.end(); ++SelectedElement){

if(SelectedElement == Spisok.begin()){

(\*SelectedElement).Save(&file\_w);

}

else{

file\_w.operator <<(endl);

(\*SelectedElement).Save(&file\_w);

}

}

file\_w.close();

}

## listreports.cpp

#include "report.h"

ListReports::ListReports(){

}

ListReports::ListReports(ifstream \*file)

{

string N;

string last;

getline(\*file, N, ' ');

Name = N;

getline(\*file, last);

int count = 1;

while ((file->eof() != true) && (count <= stoi(last)))

{

ListProducts\* NewListProducts = new ListProducts(file);

ListReports::AddElement(\*NewListProducts);

delete NewListProducts;

count++;

}

}

void ListReports::shou(){

cout << Name;

}

void ListReports::shoulist(){

cout << endl;

list<ListProducts>::iterator ItrForShouing;

ItrForShouing = Spisok.begin();

for (ItrForShouing = Spisok.begin(); ItrForShouing != Spisok.end(); ++ItrForShouing){

ItrForShouing->shou();

if (ItrForShouing == SelectedElement){

cout << " <---";

}

cout << endl;

}

}

void ListReports::AddElement(ListProducts x){

Spisok.push\_back(x);

}

void ListReports::Control(ListRecepi\* ListOfRecepi,ListIngredients\* sklad){

SelectedElement = Spisok.begin();

bool conec = true;

while (conec) {

system("cls");

ListReports::shou();

ListReports::shoulist();

switch (\_getch()){

case 80: //V

if (SelectedElement != Spisok.end())

{

SelectedElement++;

}

if (SelectedElement == Spisok.end())

{

SelectedElement--;

}

break;

case 72: // ^

if (SelectedElement != Spisok.begin())

{

SelectedElement--;

}

break;

case 13:

SelectedElement->Control(ListOfRecepi,sklad);

break;

case 27: // esc

conec = false;

break;

default:{}

}

}

}

ListProducts\* ListReports::Check(string N){

list<ListProducts>::iterator ItrToFind;

ItrToFind = Spisok.begin();

for (ItrToFind = Spisok.begin(); ItrToFind != Spisok.end(); ++ItrToFind)

if (ItrToFind->GetName() == N){

return ItrToFind->GetPointr();

};

return nullptr;

}

void ListReports::Save(){

ofstream file\_w("D:/WorkFiles/listofproducts.txt", ios\_base::out | ios\_base::trunc);

file\_w << Name + ' ';

if(Spisok.size() == 0){

file\_w << Spisok.size();

}

else{

file\_w << Spisok.size() << endl;

}

for(SelectedElement = Spisok.begin(); SelectedElement != Spisok.end(); ++SelectedElement){

if(SelectedElement == Spisok.begin()){

(\*SelectedElement).Save(&file\_w);

}

else{

file\_w.operator <<(endl);

(\*SelectedElement).Save(&file\_w);

}

}

file\_w.close();

}

## listworkers.cpp

#include "listworkers.h"

ListWorkers::ListWorkers(){

}

ListWorkers::ListWorkers(ifstream \*file)

{

string N;

string last;

getline(\*file, N, ' ');

Name = N;

getline(\*file, last);

int count = 1;

while ((file->eof() != true) && (count <= stoi(last)))

{

Worker\* NewListWorkers = new Worker(file);

ListWorkers::AddElement(\*NewListWorkers);

delete NewListWorkers;

count++;

}

}

void ListWorkers::shou(){

cout << Name;

}

void ListWorkers::shoulist(){

cout << endl;

list<Worker>::iterator ItrForShouing;

ItrForShouing = Spisok.begin();

for (ItrForShouing = Spisok.begin(); ItrForShouing != Spisok.end(); ++ItrForShouing){

ItrForShouing->shou();

if (ItrForShouing == SelectedElement){

cout << " <---";

}

cout << endl;

}

}

void ListWorkers::AddElement(Worker x){

Spisok.push\_back(x);

}

void ListWorkers::Control(){

SelectedElement = Spisok.begin();

bool conec = true;

while (conec) {

system("cls");

cout << "d - delete worker " << endl;

cout << "a - add worker " << endl;

ListWorkers::shou();

ListWorkers::shoulist();

switch (\_getch()){

case 80: //V

if (SelectedElement != Spisok.end())

{

SelectedElement++;

}

if (SelectedElement == Spisok.end())

{

SelectedElement--;

}

break;

case 72: // ^

if (SelectedElement != Spisok.begin())

{

SelectedElement--;

}

break;

case 'd':

if(!Spisok.empty() && SelectedElement->GetRoot() != "admin"){

Spisok.erase(SelectedElement);

SelectedElement = Spisok.begin();

}

break;

case 'a':{

Worker\* NewListWorkers = new Worker;

Worker\* TestListWorkers = ListWorkers::Check(NewListWorkers->GetName());

if (TestListWorkers == nullptr && NewListWorkers->GetPassword() != "" && NewListWorkers->GetRoot() != ""){

ListWorkers::AddElement(\*NewListWorkers);

}else{

cout << "that user already exist";

\_getch();

}

delete NewListWorkers;

break;}

case 27: // esc

conec = false;

break;

default:{}

}

}

}

Worker\* ListWorkers::Check(string N){

list<Worker>::iterator ItrToFind;

ItrToFind = Spisok.begin();

for (ItrToFind = Spisok.begin(); ItrToFind != Spisok.end(); ++ItrToFind)

if (ItrToFind->GetName() == N){

return ItrToFind->GetPointr();

};

return nullptr;

}

void ListWorkers::Save(){

ofstream file\_w("D:/WorkFiles/listofworkers.txt", ios\_base::out | ios\_base::trunc);

file\_w << Name + ' ';

if(Spisok.size() == 0){

file\_w << Spisok.size();

}

else{

file\_w << Spisok.size() << endl;

}

for(SelectedElement = Spisok.begin(); SelectedElement != Spisok.end(); ++SelectedElement){

if(SelectedElement == --Spisok.end()){

file\_w << (\*SelectedElement).GetName() + ' ';

file\_w << (\*SelectedElement).GetPassword() + ' ';

file\_w << (\*SelectedElement).GetRoot();

}

else{

file\_w << (\*SelectedElement).GetName() + ' ';

file\_w << (\*SelectedElement).GetPassword() + ' ';

file\_w << (\*SelectedElement).GetRoot() << endl;

}

}

file\_w.close();

}

## product.cpp

#include "product.h"

Product::Product()

{

string n, temp\_v;

char mas[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

unsigned int count = 0;

cout << "Vvedite naimenovanie: " << endl;

cin >> n;

\_flushall();

Name = n;

cout << "Vvedite kol-vo: " << endl;

cin >> temp\_v;

\_flushall();

for(unsigned int i = 0; i < temp\_v.size(); ++i){

bool exists = false;

for (int j: mas){

if (j == temp\_v[i]) {

exists = true;

break;

}

}

if(exists){

count++;

}

}

if(count == temp\_v.size()){

Value = stoi(temp\_v);

}

else{

temp\_v = "-1";

Value = stoi(temp\_v);

cout << "Vy vveli nekorrektnoe chislo!!!" << endl;

system("pause");

}

}

Product::Product(ifstream \*file)

{

string N;

string temp\_V;

getline(\*file, N, ' ');

getline(\*file, temp\_V);

Name = N;

Value = stoi(temp\_V);

}

string Product::GetName()

{

return Name;

}

int Product::GetValue()

{

return Value;

}

void Product::IncValue(int value){

this->Value += value;

}

void Product::shou()

{

cout << Name << " " << Value << " " ;

}

Product\* Product::GetPointr(){ //итератор сам не даёт указатель

return this;

}

## recepi.cpp

#include "resepi.h"

ListIngredientForRecepi::ListIngredientForRecepi(){

cout << "Vvedite imya:";

cin >> Name;

}

ListIngredientForRecepi::ListIngredientForRecepi(ifstream \*file)

{

string N;

string last;

getline(\*file, N, ' ');

Name = N;

getline(\*file, last);

int count = 1;

while ((file->eof() != true) && (count <= stoi(last)))

{

IngredientForRecepi\* NewIngredientForRecepi = new IngredientForRecepi(file);

ListIngredientForRecepi::AddElement(\*NewIngredientForRecepi);

delete NewIngredientForRecepi;

count++;

}

}

void ListIngredientForRecepi::shou(){

cout << Name;

}

void ListIngredientForRecepi::shoulist(){

cout << endl;

list<IngredientForRecepi>::iterator ItrForShouing;

ItrForShouing = Spisok.begin();

for (ItrForShouing = Spisok.begin(); ItrForShouing != Spisok.end(); ++ItrForShouing){

ItrForShouing->shou();

if (ItrForShouing == SelectedElement){

cout << " <---";

}

cout << endl;

}

}

void ListIngredientForRecepi::AddElement(IngredientForRecepi x){

Spisok.push\_back(x);

}

string ListIngredientForRecepi::GetName(){

return this->Name;

}

void ListIngredientForRecepi::Control(ListIngredients\* sklad, string root){

SelectedElement = Spisok.begin();

bool conec = true;

while (conec) {

system("cls");

if(root == "admin"){

cout << "d - delete ingredient " << endl;

cout << "a - add ingredient " << endl;

}

ListIngredientForRecepi::shou();

ListIngredientForRecepi::shoulist();

switch (\_getch()){

case 80: //V

if (SelectedElement != Spisok.end())

{

SelectedElement++;

}

if (SelectedElement == Spisok.end())

{

SelectedElement--;

}

break;

case 72: // ^

if (SelectedElement != Spisok.begin())

{

SelectedElement--;

}

break;

case 'd':

if(!Spisok.empty() && root == "admin"){

Spisok.erase(SelectedElement);

SelectedElement = Spisok.begin();

}

break;

case 'a':{if(root == "admin"){

IngredientForRecepi\* NewIngredientForRecepi = new IngredientForRecepi(1);

IngredientForRecepi\* TestIngredientForRecepi = ListIngredientForRecepi::Check(NewIngredientForRecepi->GetName());

if(NewIngredientForRecepi->GetValue() != -1 && Proverka(NewIngredientForRecepi, sklad)){

if (TestIngredientForRecepi == nullptr){

NewIngredientForRecepi->SetED((sklad->Check(NewIngredientForRecepi->GetName()))->GetED());

ListIngredientForRecepi::AddElement(\*NewIngredientForRecepi);

}

else if(TestIngredientForRecepi != nullptr){

TestIngredientForRecepi->SetValue(TestIngredientForRecepi->GetValue() + NewIngredientForRecepi->GetValue());

}

else if(!Proverka(NewIngredientForRecepi, sklad)){

cout << "nelzya dobavit ingredient v recept (net ingredienta na sklade)";

\_getch();

}

}

else{

cout << "neverno vvedeny dannye";

\_getch();

}

delete NewIngredientForRecepi;}

break;}

case 27: // esc

conec = false;

break;

default:{}

}

}

}

IngredientForRecepi\* ListIngredientForRecepi::Check(string N){

list<IngredientForRecepi>::iterator ItrToFind;

ItrToFind = Spisok.begin();

for (ItrToFind = Spisok.begin(); ItrToFind != Spisok.end(); ++ItrToFind)

if (ItrToFind->GetName() == N){

return ItrToFind->GetPointr();

};

return nullptr;

}

ListIngredientForRecepi\* ListIngredientForRecepi::GetPointr(){

return this;

}

list<IngredientForRecepi> ListIngredientForRecepi::GetSpisok(){

return this->Spisok;

}

void ListIngredientForRecepi::Save(ofstream\* file\_w){

for(unsigned int i = 0; i < Name.size(); ++i){

file\_w->put(Name[i]);

}

file\_w->put(' ');

if(Spisok.size() == 0){

file\_w->put('0');

}

else{

string str = to\_string(Spisok.size());

for(unsigned int i = 0; i < str.size(); ++i){

file\_w->put(str[i]);

}

file\_w->operator <<(endl);

}

for(SelectedElement = Spisok.begin(); SelectedElement != Spisok.end(); ++SelectedElement){

if(SelectedElement == --Spisok.end()){

string str = (\*SelectedElement).GetName();

for(unsigned int i = 0; i < str.size(); ++i){

file\_w->put(str[i]);

}

file\_w->put(' ');

string str1 = to\_string((\*SelectedElement).GetValue());

for(unsigned int i = 0; i < str1.size(); ++i){

file\_w->put(str1[i]);

}

file\_w->put(' ');

string str2 = (\*SelectedElement).GetED();

for(unsigned int i = 0; i < str2.size(); ++i){

file\_w->put(str2[i]);

}

}

else{

string str = (\*SelectedElement).GetName();

for(unsigned int i = 0; i < str.size(); ++i){

file\_w->put(str[i]);

}

file\_w->put(' ');

string str1 = to\_string((\*SelectedElement).GetValue());

for(unsigned int i = 0; i < str1.size(); ++i){

file\_w->put(str1[i]);

}

file\_w->put(' ');

string str2 = (\*SelectedElement).GetED();

for(unsigned int i = 0; i < str2.size(); ++i){

file\_w->put(str2[i]);

}

file\_w->operator <<(endl);

}

}

}

bool ListIngredientForRecepi::Proverka(IngredientForRecepi\* ptr\_ingredient\_for\_recepi, ListIngredients\* ptr\_list\_ingredients){

if(ptr\_list\_ingredients->Check(ptr\_ingredient\_for\_recepi->GetName()) == nullptr){

return false;

}

else{

return true;

}

}

## report.cpp

#include "listproducts.h"

ListProducts::ListProducts(){

}

ListProducts::ListProducts(string N){

Name = N;

}

ListProducts::ListProducts(ifstream \*file)

{

string N;

string last;

getline(\*file, N, ' ');

Name = N;

getline(\*file, last);

int count = 1;

while ((file->eof() != true) && (count <= stoi(last)))

{

Product\* NewListProducts = new Product(file);

ListProducts::AddElement(\*NewListProducts);

delete NewListProducts;

count++;

}

}

void ListProducts::shou(){

cout << Name;

}

void ListProducts::shoulist(){

cout << endl;

list<Product>::iterator ItrForShouing;

ItrForShouing = Spisok.begin();

for (ItrForShouing = Spisok.begin(); ItrForShouing != Spisok.end(); ++ItrForShouing){

ItrForShouing->shou();

if (ItrForShouing == SelectedElement){

cout << " <---";

}

cout << endl;

}

}

void ListProducts::AddElement(Product x){

Spisok.push\_back(x);

}

string ListProducts::GetName(){

return this->Name;

}

void ListProducts::Control(ListRecepi\* ListOfRecepi,ListIngredients\* sklad){

SelectedElement = Spisok.begin();

time\_t t = time(nullptr);

tm\* now = localtime(&t);

string today = to\_string(now->tm\_mday) + '.' + to\_string((now->tm\_mon + 1)) + '.' + to\_string((now->tm\_year + 1900));

bool conec = true;

while (conec) {

system("cls");

if(this->Name == today){

cout << "a - add product " << endl;}

ListProducts::shou();

ListProducts::shoulist();

switch (\_getch()){

case 80: //V

if (SelectedElement != Spisok.end())

{

SelectedElement++;

}

if (SelectedElement == Spisok.end())

{

SelectedElement--;

}

break;

case 72: // ^

if (SelectedElement != Spisok.begin())

{

SelectedElement--;

}

case 'd':

break;

case 'a':{

if(this->Name == today){

Product\* NewListProducts = new Product;

Product\* TestListProducts = ListProducts::Check(NewListProducts->GetName());

if (NewListProducts->GetValue() != -1 && Proverka(NewListProducts, ListOfRecepi, sklad)){

if(TestListProducts == nullptr){

ListProducts::AddElement(\*NewListProducts);

}

else{

TestListProducts->IncValue(NewListProducts->GetValue());

}

}

else if(!Proverka(NewListProducts, ListOfRecepi, sklad)){

cout << "Nelzya dobavit produkt v otchet (ne hvataet ingredientov na sklade)" << endl;

system("pause");

}

delete NewListProducts;

}

else{

cout << "Mozhno izmenyat tolko za tekushchij den!";

system("pause");

}

break;}

case 27: // esc

conec = false;

break;

default:{}

}

}

}

Product\* ListProducts::Check(string N){

list<Product>::iterator ItrToFind;

ItrToFind = Spisok.begin();

for (ItrToFind = Spisok.begin(); ItrToFind != Spisok.end(); ++ItrToFind)

if (ItrToFind->GetName() == N){

return ItrToFind->GetPointr();

};

return nullptr;

}

ListProducts\* ListProducts::GetPointr(){

return this;

}

void ListProducts::Save(ofstream\* file\_w){

for(unsigned int i = 0; i < Name.size(); ++i){

file\_w->put(Name[i]);

}

file\_w->put(' ');

if(Spisok.size() == 0){

file\_w->put('0');

}

else{

string str = to\_string(Spisok.size());

for(unsigned int i = 0; i < str.size(); ++i){

file\_w->put(str[i]);

}

file\_w->operator <<(endl);

}

for(SelectedElement = Spisok.begin(); SelectedElement != Spisok.end(); ++SelectedElement){

if(SelectedElement == --Spisok.end()){

string str = (\*SelectedElement).GetName();

for(unsigned int i = 0; i < str.size(); ++i){

file\_w->put(str[i]);

}

file\_w->put(' ');

string str1 = to\_string((\*SelectedElement).GetValue());

for(unsigned int i = 0; i < str1.size(); ++i){

file\_w->put(str1[i]);

}

}

else{

string str = (\*SelectedElement).GetName();

for(unsigned int i = 0; i < str.size(); ++i){

file\_w->put(str[i]);

}

file\_w->put(' ');

string str1 = to\_string((\*SelectedElement).GetValue());

for(unsigned int i = 0; i < str1.size(); ++i){

file\_w->put(str1[i]);

}

file\_w->operator <<(endl);

}

}

}

bool ListProducts::Proverka(Product\* ptr\_product, ListRecepi\* ptr\_list\_recepi, ListIngredients\* ptr\_list\_ingredient){

ListIngredientForRecepi\* recept = ptr\_list\_recepi->Check(ptr\_product->GetName());

unsigned count = 0;

if(recept != nullptr){ //ввели правильный продукт(рецепт), такой есть в списке рецептов

list<IngredientForRecepi> Spisok = recept->GetSpisok();

list<IngredientForRecepi>::iterator SelectedElement;

for(SelectedElement = Spisok.begin(); SelectedElement != Spisok.end(); ++SelectedElement){

if(ptr\_list\_ingredient->Check((\*SelectedElement).GetName()) != nullptr){ //есть ингредиент из рецепта на складе

if(ptr\_list\_ingredient->Check((\*SelectedElement).GetName())->GetValue() >= (ptr\_product->GetValue() \* (\*SelectedElement).GetValue())){//на складе достаточно ингредиентов для добавления готовой продукции

++count;

}

}

}

if(count == Spisok.size()){

for(SelectedElement = Spisok.begin(); SelectedElement != Spisok.end(); ++SelectedElement){

ptr\_list\_ingredient->Check((\*SelectedElement).GetName())->SetValue(ptr\_list\_ingredient->Check((\*SelectedElement).GetName())->GetValue() - ((ptr\_product->GetValue() \* (\*SelectedElement).GetValue())));

}

return true;

}

else{

return false;

}

}

else{

return false;

}

}

## userinterface.cpp

#include "userinterface.h"

using namespace std;

UserInterface::UserInterface()

{

}

void UserInterface::Login(){

ifstream file;

file.open("D:/WorkFiles/listofworkers.txt", ios\_base::in);

ListWorkers WorkerList(&file);

string login;

string password;

bool exit = false;

do{

cout << "Login: ";

cin >> login;

Worker\* buff = WorkerList.Check(login);

if(buff == nullptr){

cout << "Takogo polzovatelya ne sushchestvuet!!" << endl;

}

else{

cout << "Parol: ";

cin >> password;

if(buff->GetPassword() == password){

switch (buff->GetRoot()[0]) {

case 'a':

UserInterface::Admin();

exit = true;

break;

case 'p':

UserInterface::Pekar();

exit = true;

break;

case 'r':

UserInterface::Sklader();

exit = true;

break;

}

}

else{

cout << "Vveden nevernyj parol polzovatelya!!" << endl;

}

}

}while(!exit);

}

void UserInterface::Admin(){

ifstream fileW;

fileW.open("D:/WorkFiles/listofworkers.txt", ios\_base::in);

ListWorkers WorkerList(&fileW);

fileW.close();

ifstream fileS;

fileS.open("D:/WorkFiles/sklad.txt", ios\_base::in);

ListIngredients Sklad(&fileS);

fileS.close();

ifstream fileR;

fileR.open("D:/WorkFiles/listofrecepi.txt", ios\_base::in);

ListRecepi ListOfRecepi(&fileR);

fileR.close();

ifstream fileP;

fileP.open("D:/WorkFiles/listofproducts.txt", ios\_base::in);

ListReports ListOfReports(&fileP);

fileP.close();

time\_t t = time(nullptr);

tm\* now = localtime(&t);

string today = to\_string(now->tm\_mday) + '.' + to\_string((now->tm\_mon + 1)) + '.' + to\_string((now->tm\_year + 1900));

if (ListOfReports.Check(today) == nullptr)

{

ListProducts\* NewListProducts = new ListProducts(today);

ListOfReports.AddElement(\*NewListProducts);

delete NewListProducts;

}

string root = "admin";

bool conec = true;

while (conec) {

system("cls");

cout << "W - workers " << endl;

cout << "S - sklad " << endl;

cout << "R - recepi " << endl;

cout << "O - otchety " << endl;

switch (\_getch()){

case 'w':

WorkerList.Control();

break;

case 's':

Sklad.Control(root);

break;

case 'r':

ListOfRecepi.Control(&Sklad, root);

break;

case 'o':

ListOfReports.Control(&ListOfRecepi,&Sklad);

break;

case 27: // esc

conec = false;

WorkerList.Save();

Sklad.Save();

ListOfRecepi.Save();

ListOfReports.Save();

break;

default:{}

}

}

}

void UserInterface::Pekar(){

ifstream fileS;

fileS.open("D:/WorkFiles/sklad.txt", ios\_base::in);

ListIngredients Sklad(&fileS);

fileS.close();

ifstream fileR;

fileR.open("D:/WorkFiles/listofrecepi.txt", ios\_base::in);

ListRecepi ListOfRecepi(&fileR);

fileR.close();

ifstream fileP;

fileP.open("D:/WorkFiles/listofproducts.txt", ios\_base::in);

ListReports ListOfReports(&fileP);

fileP.close();

time\_t t = time(nullptr);

tm\* now = localtime(&t);

string today = to\_string(now->tm\_mday) + '.' + to\_string((now->tm\_mon + 1)) + '.' + to\_string((now->tm\_year + 1900));

if (ListOfReports.Check(today) == nullptr)

{

ListProducts\* NewListProducts = new ListProducts(today);

ListOfReports.AddElement(\*NewListProducts);

delete NewListProducts;

}

string root = "pekar";

bool conec = true;

while (conec) {

system("cls");

cout << "S - sklad " << endl;

cout << "R - recepi " << endl;

cout << "O - otchet " << endl;

switch (\_getch()){

case 's':

Sklad.Control(root);

break;

case 'r':

ListOfRecepi.Control(&Sklad, root);

break;

case 'o':

ListOfReports.Check(today)->Control(&ListOfRecepi,&Sklad);

break;

case 27: // esc

conec = false;

Sklad.Save();

ListOfRecepi.Save();

ListOfReports.Save();

break;

default:{}

}

}

}

void UserInterface::Sklader(){

ifstream fileS;

fileS.open("D:/WorkFiles/sklad.txt", ios\_base::in);

ListIngredients Sklad(&fileS);

fileS.close();

string root = "rabochiy";

Sklad.Control(root);

Sklad.Save();

}

## worker.cpp

#include "worker.h"

Worker::Worker()

{

string temp\_password, temp\_root;

char mas[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

unsigned int count = 0;

cout << "Vvedite login: " << endl;

cin >> this->Login;

cout << "Vvedite parol: " << endl;

cin >> temp\_password;

\_flushall();

for(unsigned int i = 0; i < temp\_password.size(); ++i){

bool exists = false;

for (int j: mas){

if (j == temp\_password[i]) {

exists = true;

break;

}

}

if(exists){

count++;

}

}

if(count == temp\_password.size()){

Password = temp\_password;

cout << "Neobhodimo ukazat prava dostupa." << endl;

cout << "Prava dostupa: 1 - pekar, 2 - rabochiy.(ukazhite cifru)" << endl;

cout << "Prava dostupa: ";

cin >> temp\_root;

\_flushall();

if(temp\_root == "1"){

Root = "pekar";

}

else if(temp\_root == "2"){

Root = "rabochiy";

}

else{

Root = "";

cout << "Vy vveli nekorrektnoe chislo!!!" << endl;

system("pause");

}

}

else{

Password = "";

Root = "";

cout << "Vy vveli nekorrektnij parol!!!" << endl;

system("pause");

}

}

Worker::Worker(ifstream \*file)

{

string L;

string P;

string R;

getline(\*file, L, ' ');

getline(\*file, P, ' ');

getline(\*file, R);

Login = L;

Password = P;

Root = R;

}

string Worker::GetName()

{

return Login;

}

string Worker::GetPassword()

{

return Password;

}

string Worker::GetRoot(){

return Root;

}

void Worker::shou()

{

cout << Login << " " << Password << " " << Root;

}

Worker\* Worker::GetPointr(){ //итератор сам не даёт указатель

return this;

}

## main.cpp

#include <iostream>

#include <string>

#include "userinterface.h"

using namespace std;

int main()

{

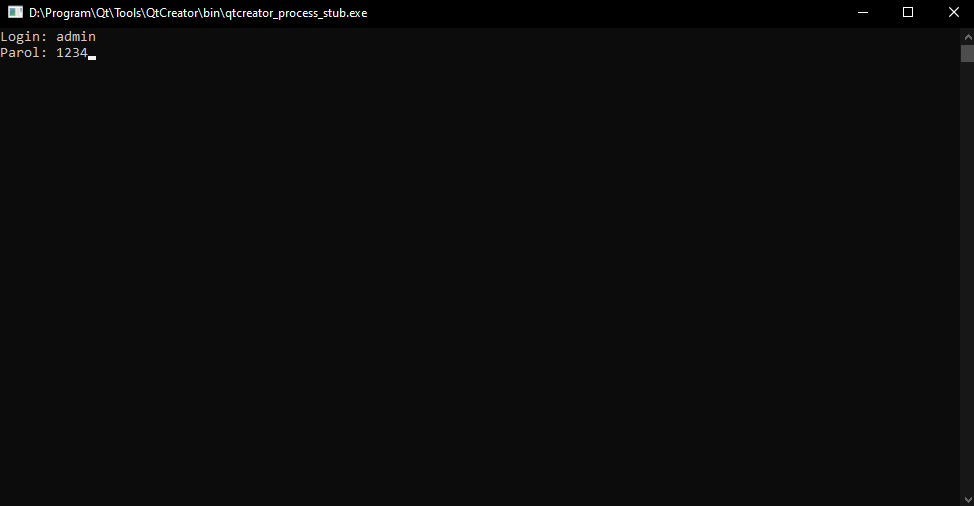
UserInterface the\_user\_interface;

the\_user\_interface.Login();

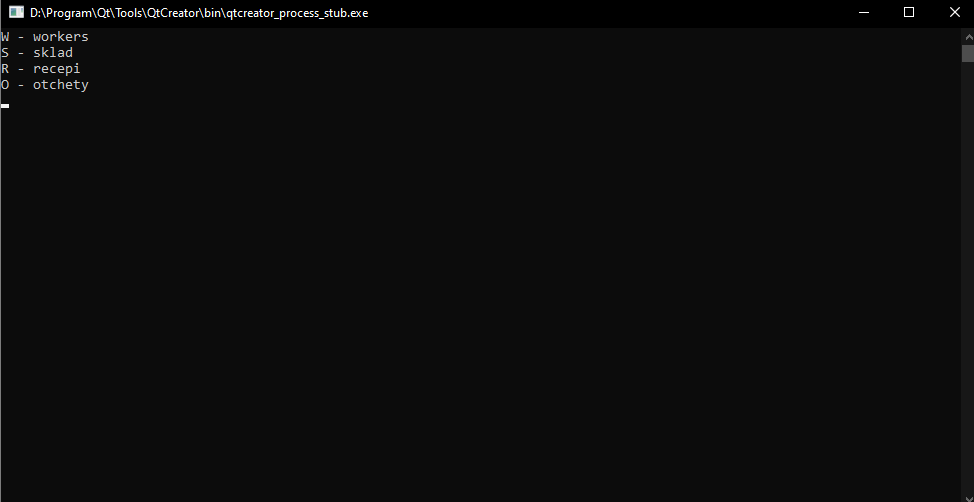
}

**Руководство пользователя**

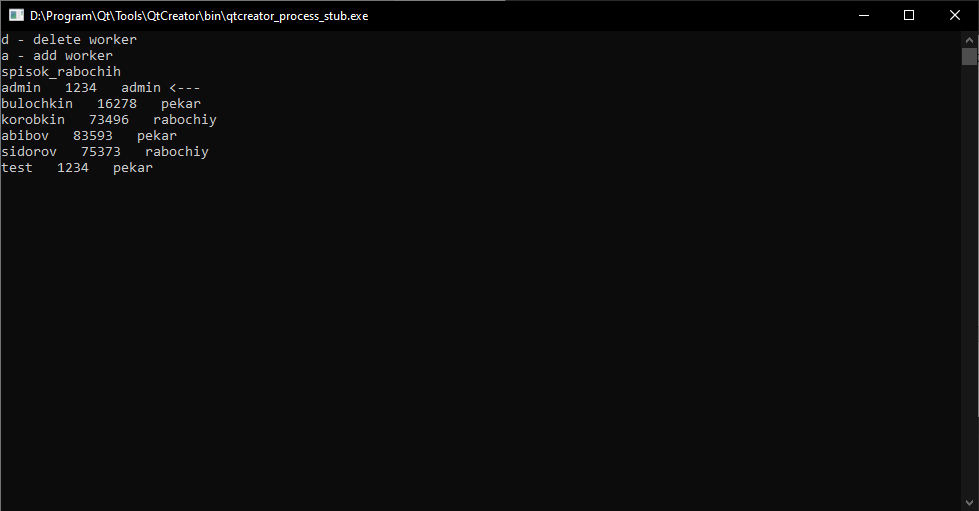
1. При запуске программы выводится регистрационное окно, в котором необходимо указать свой личный логин и пароль:



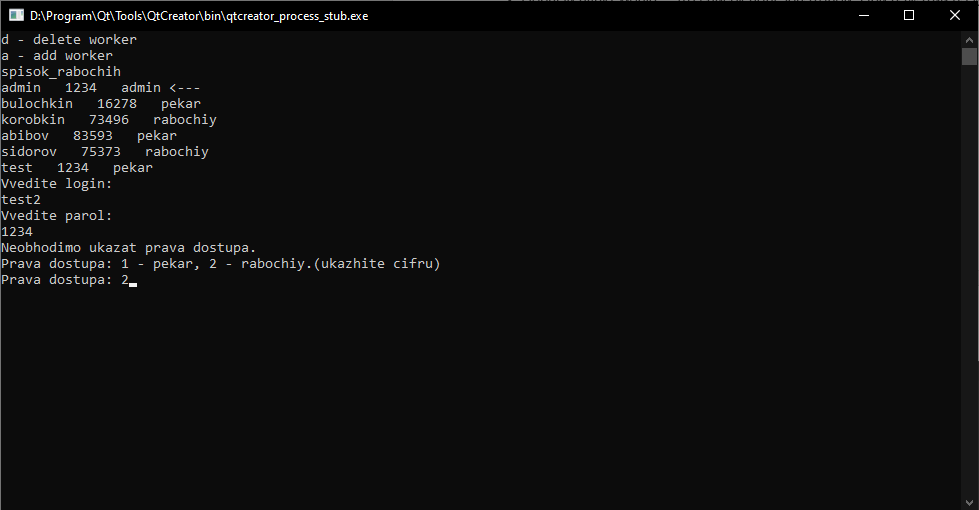
2. После ввода правильного ввода логина и пароля пользователь попадает в главное меню:

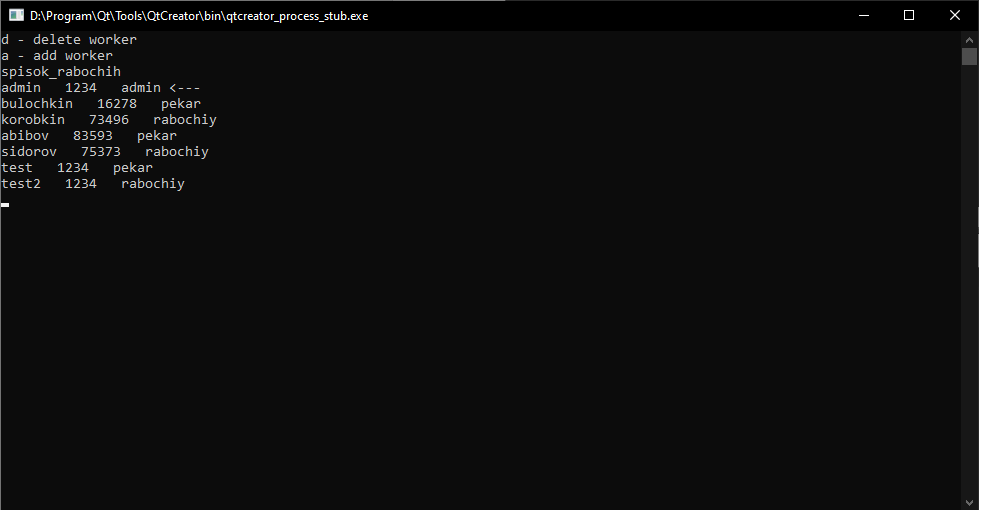


3. Первый пункт меню – это список пользователей. При нажатии на клавишу W пользователь попадет в список пользователей:

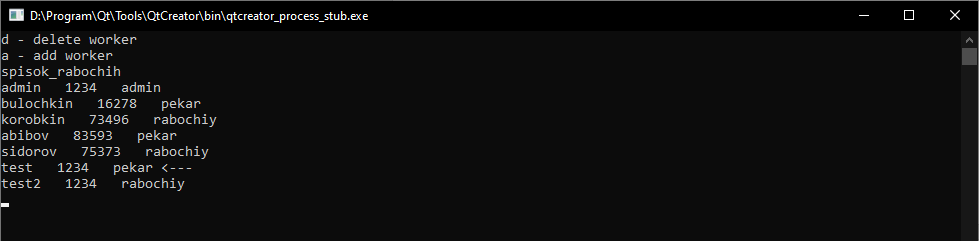


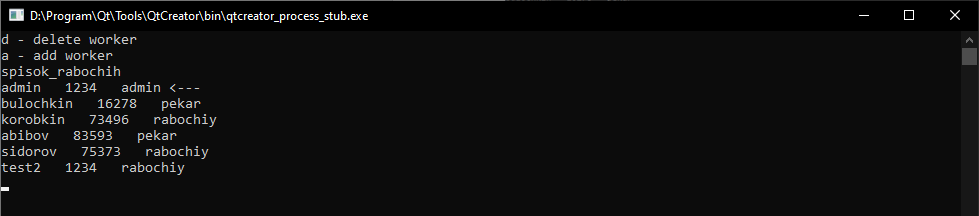
3.1. В списке пользователей на клавишу A можно добавить пользователя в систему, для этого нужно ввести данные нового пользователя:





3.2. В списке пользователей на клавишу D можно удалить пользователя из системы, для этого нужно переместить указатель на нужного пользователя (стрелками вверх и вниз):



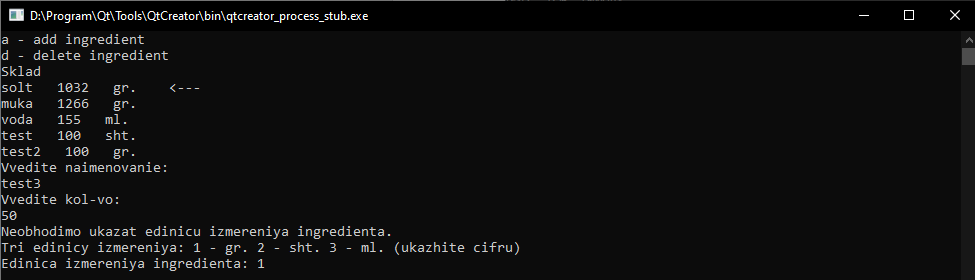


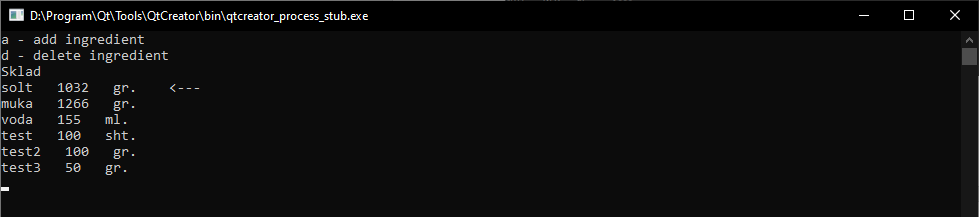
3.3. Для выхода в предыдущее меню необходимо нажать клавишу Esc.

4. Второй пункт меню – это склад. При нажатии на клавишу S пользователь попадет на склад:



4.1. На складе на клавишу A можно добавить ингредиент на склад, для этого нужно ввести данные нового ингредиента:





4.2. На складе на клавишу D можно удалить ингредиент со склада, для этого нужно переместить указатель на нужный ингредиент (стрелками вверх и вниз):



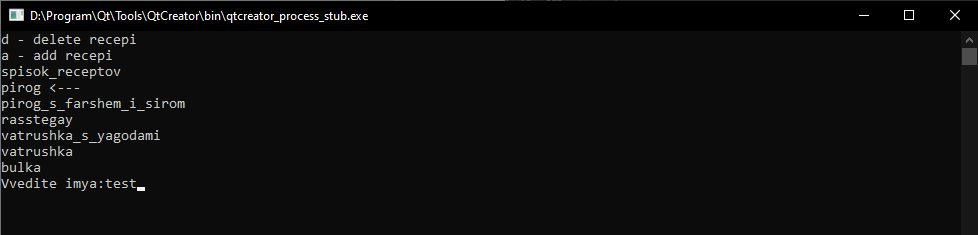


4.3. Для выхода в предыдущее меню необходимо нажать клавишу Esc.

5. Третий пункт меню – это список рецептов. При нажатии на клавишу R пользователь попадет в список рецептов:



5.1. В списке рецептов на клавишу A можно добавить рецепт, для этого нужно ввести имя нового рецепта:





5.2. В списке рецептов на клавишу D можно удалить рецепт, для этого нужно переместить указатель на нужный рецепт (стрелками вверх и вниз):





5.3. В списке рецептов на клавишу Enter можно посмотреть содержимое рецепта, для этого нужно переместить указатель на нужный рецепт (стрелками вверх и вниз):





5.3.1. В содержимом рецепта на клавишу A можно добавить ингредиент для рецепта, для этого нужно данные нового рецепта:





5.3.2. В содержимом рецепта на клавишу D можно удалить рецепт, для этого нужно переместить указатель на нужный рецепт (стрелками вверх и вниз):





5.3.3. Для выхода в предыдущее меню необходимо нажать клавишу Esc.

5.4. Для выхода в предыдущее меню необходимо нажать клавишу Esc.

6. Четвертый пункт меню – это список отчетов. При нажатии на клавишу O пользователь попадет в список отчетов:



6.1. В списке отчетов на клавишу Enter можно посмотреть содержимое отчета, для этого нужно переместить указатель на нужный рецепт (стрелками вверх и вниз):





6.1.1. В списке отчетов на клавишу A можно добавить продукт для отчета, для этого нужно данные нового продукта:





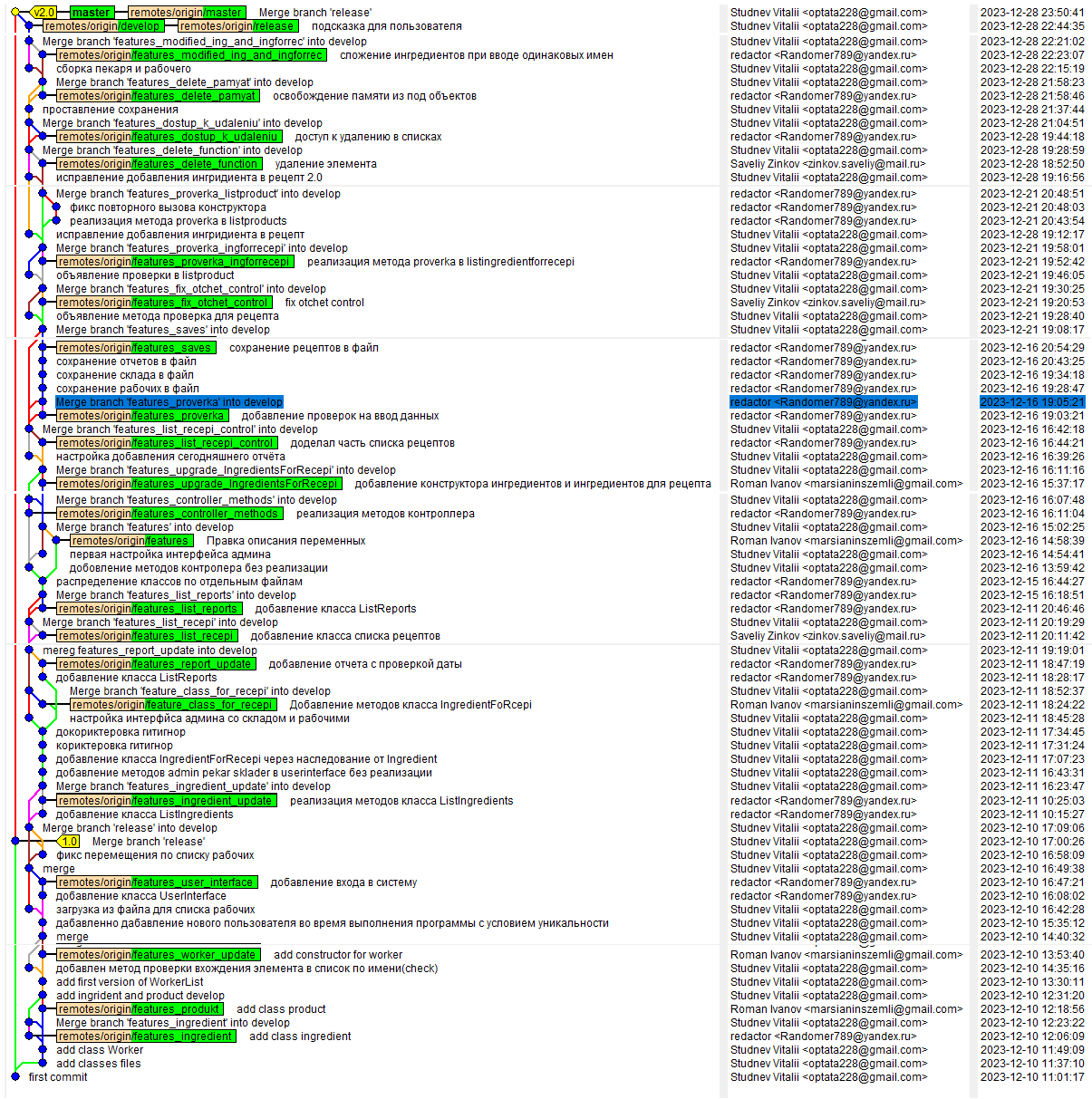
6.1.2. Для выхода в предыдущее меню необходимо нажать клавишу Esc.

6.2. Для выхода в предыдущее меню необходимо нажать клавишу Esc.

7. Для выхода из программы необходимо нажать клавишу Esc.

Инструкция выше написана для администратора. В системе имеются еще пекарь и работник склада, их функционал программы ограничен их правами доступа. Работник склада может только смотреть список ингредиентов на складе и добавлять ингредиенты на склад. Пекарь может смотреть список ингредиентов на складе, смотреть список рецептов и содержимое каждого рецепта, смотреть содержимое отчета за текущий день и добавлять продукцию в этот отчет.

**История проекта на GitHub**



Адрес проекта: https://github.com/StudnevVitalii/techprog\_TeamProject.git

**Заключение:**

Для создания проекта была использована среда разработки Qt Creator 5.4.2. Программа написана на языке программирования C++. Применялась система контроля версий Git. Все прецеденты реализованы, сбои и зависания не наблюдаются, использованы принципы раздельной компиляции, приложены диаграммы: вариантов использования, действий, классов, последовательностей.