Частное учреждение образования

«Колледж бизнеса и права»

web-сайт «IT Pragmatika»

Пояснительная Записка

к дипломному проекту

КП Т.594008.401

Руководитель проекта ( Н.И. Белугина )

Учащийся ( В.А. Евсейчиков )

2019

Содержание

[Введение 4](#_Toc534671283)

[1 Объектно-ориентированный анализ и проектирование системы 5](#_Toc534671284)

[1.1 Сущность задачи 5](#_Toc534671285)

[1.2 Проектирование модели 6](#_Toc534671286)

[2 Вычислительная система 9](#_Toc534671287)

[2.1 Требования к аппаратным и операционным ресурсам 9](#_Toc534671288)

[2.2 Инструменты разработки 9](#_Toc534671289)

[3 Проектирование задачи 10](#_Toc534671290)

[3.1 Требования к приложению 10](#_Toc534671291)

[3.2 Концептуальный прототип 10](#_Toc534671292)

[3.3 Организация данных 11](#_Toc534671293)

[3.4 Функции и элементы управления 13](#_Toc534671294)

[3.5 Проектирование справочной системы приложения 15](#_Toc534671295)

[4 Описание программного средства 16](#_Toc534671296)

[4.1 Общие сведения 16](#_Toc534671297)

[4.2 Функциональное назначение 16](#_Toc534671298)

[4.3 Входные и выходные данные 16](#_Toc534671299)

[5 Методика испытаний 18](#_Toc534671300)

[5.1 Технические требования 18](#_Toc534671301)

[5.2 Функциональное тестирование 18](#_Toc534671302)

[6 Применение 24](#_Toc534671303)

[6.1 Назначение программы 24](#_Toc534671304)

[6.2 Условия применения 24](#_Toc534671305)

[6.3 Справочная система 24](#_Toc534671306)

Заключение 27

Список информационных источников 28

Приложение А Текст программных модулей 29

у

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

КП Т.594008.401 ПЗ

Разраб.

Евсейчиков В.А.

Провер.

Белугина Н.И.

Реценз.

Н. Контр.

Утверд.

Web-сайт «IT Pragmatika»

Лит.

Листов

45

КБП

У

Введение

Во время развития компьютерных технологий, хранение информации на бумаге становится все более и более невостребованным, так как информация на бумаге занимает много места, анализ этой информации требует большого количества усилий и времени и поиск нужного документа также становится трудоемким.

С целью облегчения и повышения эффективности работы возникла необходимость в использовании современных web - сайтов.

Целью курсового проекта является разработка web-сайта «IT Pragmatika», которое будет осуществлять ведение базы данных о клиентах, содержание информации о компании и предоставляемых услугах.

Пояснительная записка состоит из шести разделов, содержащих необходимую информацию по организации эксплуатации программного средства.

В первом разделе «Объектно-ориентированный анализ и проектирование системы» раскрывается организационная сущность задачи, описывается предметная область и круг задач, которые должны быть автоматизированы. Описывается задача, перечисляются основные функции программы. Строится информационная модель, отражающая сущности задачи, их свойства и взаимосвязи.

Во втором разделе «Вычислительная система» перечисляются требования к аппаратному и программному обеспечению компьютера, проводится характеристика операционной системы, обоснование выбранной среды для разработки програмы.

В третьем разделе «Проектирование задачи» проводится объектно-ориентированный анализ задачи, приводится требования к приложению, концептуальный прототип, организация данных, функции и элементы управления, проектирование справочной системы.

В четвертом разделе «Описание программного средства» представлены общие сведения о программном средстве и его функциональном назначении.

В пятом разделе «Методика испытаний» описываются требования к аппаратному и программному обеспечению компьютера для проведения испытаний, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Представляются результаты функционального тестирования.

Шестой раздел «Применение» предназначен для описания сведений о назначении программного средства и области его применения. В этом разделе приводится структура справочной системы, а также методика ее использования.

В заключении анализируется созданное программное средство, определяется степень соответствия поставленной задачи и выполненной работы.

Приложение А содержит текст программных модулей.

В графической части представлены диаграммы вариантов использования, классов, деятельности, последовательности и компонентов.

1. Объектно-ориентированный анализ и проектирование системы
   1. Сущность задачи

Предметной областью решаемой задачи является компания «Pragmatika», которая предоставляет необходимую информацию о компании и ее услугах, а так же автоматизирует процесс взаимодействия с пользователем, путем создания формы обратной связи и возможности оставления комментариев.

Исходя из анализа предметной области можно выделить следующие задачи, подлежащие автоматизации:

* ведение базы данных (БД) о пользователях;
* взаимодействие с пользователем, путем создания формы обратной связи;
* возможность просмотра услуг и их стоимости, предоставляемых компанией;
* возможность просмотра галереи работ, проделанных компанией;
* возможность авторизации/регистрации пользователей;
* возможность оставления комментариев;
* разработка двуязычного сайта (русская и английская версии).

Существует много аналогов web-сайтов, которые автоматизируют процесс взаимодействия с пользователем. Аналогом web-сайта является компания «Itechart».

* 1. Проектирование модели

Данный программный модуль будет автоматизировать процесс взаимодействия сайта с пользователем.

Цель моделирования данных состоит в обеспечении разработчика информационной системы концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются диаграммы «Сущность-связь» (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных. Нотация ERD была впервые введена П. Ченном и получила дальнейшее развитие в работах Баркера.

Диаграмма «Сущность-связь» представлена на рисунке 1.

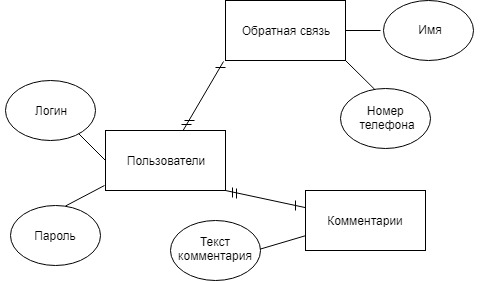


Рисунок 1 – Диаграмма «Сущность-связь»

Исходя из исследования предметной области, можно выделить следующие сущности разработки: «Пользователи», «Обратная связь», «Комментарии».

Для сущности «Пользователи» можно выделить следующие атрибуты:

* пароль;
* логин.

Для сущности «Обратная связь» можно выделить следующие атрибуты:

* имя;
* номер телефона.

Для сущности «Комментарии» можно выделить следующий атрибут:

* текст комментария.

В рамках языка UML все представления о модели сложной системы фиксируются в виде специальных графических конструкций – диаграмм. В терминах языка UML определены следующие виды диаграмм: диаграмма вариантов использования, диаграмма классов, диаграмма деятельности, диаграмма последовательности, диаграмма компонентов.

Суть диаграммы вариантов использования состоит в том, что проектируемая система представляется в виде множества сущностей или актёров, взаимодействующих с системой с помощью, так называемых, вариантов использования.

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы.

Актёр представляет собой внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует её функциональные возможности для достижения определённых целей или решения частных задач. При этом актёры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актёр может рассматриваться как некоторая отдельная роль относительно конкретного варианта использования. Суть диаграммы вариантов использования состоит в том, что проектируемая система представляется в виде множества сущностей или актёров, взаимодействующих с системой с помощью, так называемых, вариантов использования.

Данная программа имеет следующие основные функции:

* ведение базы данных;
* отправка данных с формы обратной связи;
* авторизация, регистрация;
* отправка комментариев.

К вспомогательным функциям, расширяющим возможности системы относятся следующие функции:

* просмотр информации, услуг и их стоимости компании;
* просмотр галереи работ;
* двуязычный сайт (русская и английская версии).

Диаграмма вариантов использования представлена в графической части на листе 2.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы. Диаграмма классов для проектируемой системы представлена в графической части на листе 3.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Графически диаграмма деятельности представляется в форме графа, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому. Основная цель использования диаграмм деятельности - визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности для функции поиска информации представлена в графической части на листе 5.

Для моделирования взаимодействия объектов в UML используются соответствующие диаграммы взаимодействия. Если рассматривать взаимодействия объектов во времени, тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности.

Временной аспект поведения имеет существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели и используются диаграммы последовательности, в которых ключевым моментом является динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения: одно - слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии; второе - вертикальная временная ось, направленная сверху вниз, на которой начальному моменту времени соответствует самая верхняя часть диаграммы. Диаграмма последовательности для проектируемой системы представлена в графической части на листе 4.

Рассмотренные ранее диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления. Особенность логического представления заключается в том, что оно оперирует понятиями, которые не имеют самостоятельного материального воплощения. Другими словами, различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически. Они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

Основное назначение логического представления состоит в анализе структурных и функциональных отношений между элементами модели системы. Однако для создания конкретной физической системы необходимо некоторым образом реализовать все элементы логического представления в конкретные материальные сущности. Для описания таких реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели.

1. Вычислительная система
   1. Требования к аппаратным и операционным ресурсам

Конфигурация компьютера, на котором будет разрабатываться программное приложение:

* процессор Intel Core i5 7600 3,50 Ггц;
* оперативная память DDR4 8Гбайт;
* жёсткий диск HDD Toshiba 2Тбайт;
* видеокарта AMD Radeon R7 370 4Гбайт;
* материнская плата Asus;
* монитор Viewsonic 24 дюйма;
* клавиатура, мышь.
  1. Инструменты разработки

Инструменты разработки:

* операционная система Windows 10;
* среда разработки текстовый редактор Visual Studio Code;
* язык программирования HTML5, CSS3, Java Script, PHP;
* система управления базами данных MySQL;
* программа для создания диаграмм UMLet.

Операционная система Windows 10 появилась относительно недавно – она стала доступной с 29 июля 2015 года. Компания Microsoft при разработке продолжала свой путь, направленный на унификацию. Допускается установка на компьютеры, ноутбуки, планшеты, а также смартфоны и консоли.

Visual Studio Code – текстовый редактор для кроссплатформенной разработки веб-сайтов и веб-приложений.

HTML - cтандартизированный язык разметки документов во всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

CSS - формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML.

Java Script – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

MySQL – свободная объектно-реляционная система управления базами данных.

PHP – [скриптовый язык](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) общего назначения, интенсивно применяемый для разработки веб-приложений.

UMLet – это [инструмент UML](https://en.wikipedia.org/wiki/UML_tool) с [открытым исходным кодом на](https://en.wikipedia.org/wiki/Open-source) основе [Java](https://en.wikipedia.org/wiki/Java_(programming_language))[,](https://en.wikipedia.org/wiki/UML_tool)предназначенный для обучения [унифицированному языку моделирования](https://en.wikipedia.org/wiki/Unified_Modeling_Language)и для быстрого создания диаграмм UML. Это инструмент рисования, а не инструмент моделирования, поскольку нет базового словаря или каталога объектов многократного использования. UMLet распространяется по [лицензии GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License) . UMLet имеет простой пользовательский интерфейс, который использует коды форматирования текста для изменения основных фигур с помощью украшений и аннотаций, поэтому нет леса икон или диалогов списка параметров по-своему.

1. Проектирование задачи
   1. Требования к приложению

Данное приложение не требует никаких специальных средств защиты либо ограничений прав доступа к данным.

Разрабатываемое приложение должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено. Добавление и изменение будет происходить с помощью табличной формы. Для более удобного перемещения между таблицами будут использоваться ссылки.

Все входные данные должны проверяться на ошибки. При совершении пользователем ошибки, ему будет предоставлено диалоговое окно с разъяснением ошибки.

Кроме этого при разработке форм необходимо соблюдать определенные требования: интерфейс в приложении должен иметь стандартные элементы управления данными, не должен быть перегружен информацией и понятен простому пользователю.

В создаваемом программном средстве должна быть справочная система, в которой будут предусмотрены все возможные способы работы системы. Справочная система должна содержать ответы на вопросы, которые могут возникнуть у пользователя.

* 1. Концептуальный прототип

Концептуальный прототип состоит из описания внешнего пользовательского интерфейса, а именно, элементов управления.

При создании данного приложения важную роль играют окна, так как они являются основным диалоговым средством работы пользователя.

При проектировании концептуального прототипа предполагается, что при загрузке программы первой будет загружаться окно «Блюда», макет которого представлен на рисунке 3.

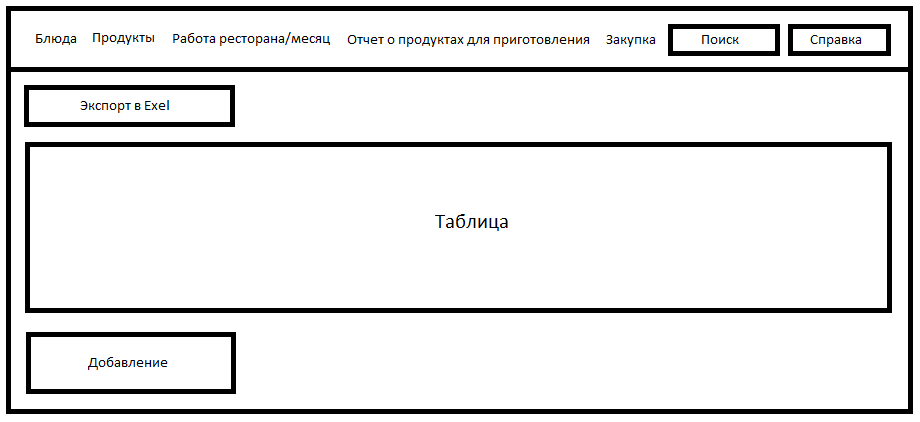


Рисунок 3 – Блюда

На ней присутствует система вкладок для работы с другими таблицами соответственно: «Блюда», «Продукты», «Работа ресторана/месяц», «Закупка». Для перехода по вкладкам, добавления используется компонент «button», для отображения данных из базы данных, используется компонент «table». Для поиска информации используется текстовое поле «Поиск». Для перехода к справке используется окно «Справка».

На окне «Добавление» располагаются текстовые поля для заполнения данных и для внесения изменений в базу данных. Макет окна «Добавление» представлен на рисунке 4.

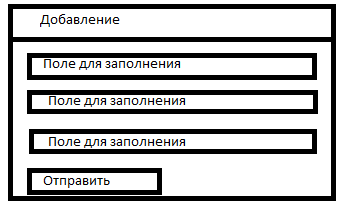


Рисунок 4 – Добавление

* 1. Организация данных

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы - атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация — это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

Структура базы данных разрабатываемого программного средства включает пять таблиц.

Структура данных таблиц, и их краткое описание приводится в таблицах 1-5.

Таблица «Category\_of\_product» хранит информацию о категориях продуктов. Структура приведена в таблице 1

Таблица 1 – Структура таблицы «Category\_of\_product»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id\_category | integer | 4 | Идентификатор категории |
| Name\_category | nvarchar | 40 | Название категории |
| Unit\_of\_measurements | varchar | 4 | Единица измерения |

Таблица «Products» хранит информацию о продуктах. Структура приведена в   
таблице 2.

Таблица 2 – Структура таблицы «Products»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id\_product | integer | 4 | Идентификатор продукта |
| Name\_product | nvarchar | 40 | Название продукта |
| Count\_available | integet | 4 | Количество товара |
| Fkey\_category | integer | 4 | Идентификатор таблицы «Category\_of\_product» |

Таблица «Purchase» хранит информацию о закупке продуктов. Структура приведена в   
таблице 3.

Таблица 3 – Структура таблицы «Purchase»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id\_purchase | integer | 4 | Идентификатор закупки |
| Price\_purchase | integer | 4 | Цена закупки |
| Date\_of\_supply | date | 4 | Дата закупки |
| Count\_of\_purchase\_products | integer | 4 | Количество закупаемых продуктов |
| Fkey\_product | integer | 4 | Идентификатор таблицы «Products» |

Таблица «Dish» хранит информацию о блюдах. Структура приведена в   
таблице 4.

Таблица 4 – Структура таблицы «Dish»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id\_dish | integer | 4 | Идентификатор блюда |
| Name\_dish | nvarchar | 40 | Название блюда |
| Price\_dish | integer | 4 | Цена блюда |

Таблица «Cooking» хранит информацию о приготовлении блюд. Структура приведена в   
таблице 5.

Таблица 5 – Структура таблицы «Cooking»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id\_cooking | integer | 4 | Идентификатор приготовления |
| Count\_of\_use\_products | integer | 4 | Количество используемых продуктов |
| Fkey\_dish | integer | 4 | Идентификатор таблицы «Dish» |
| Fkey\_product | integer | 4 | Идентификатор таблицы «Products» |

Физическая схема базы данных представлена на рисунке 5.

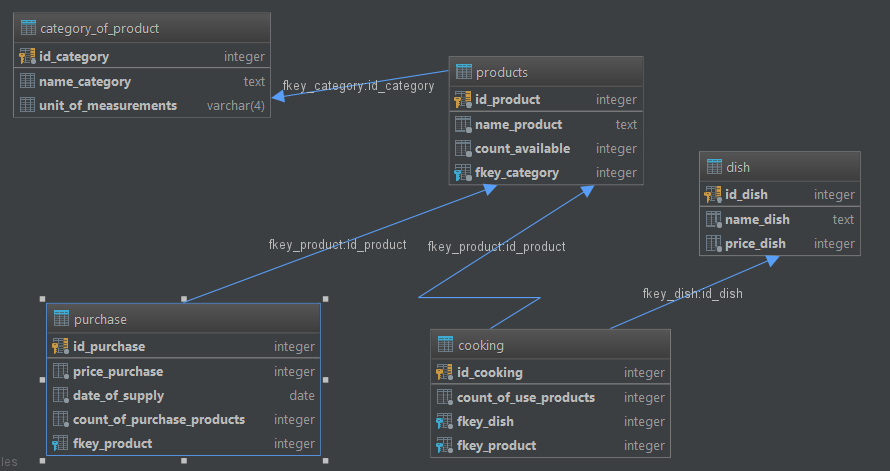


Рисунок 5 – Физическая схема базы данных

* 1. Функции и элементы управления

Основными функциями данного проекта являются добавление блюд и закупка продуктов, автоматический учет закупки продуктов за месяц.

При добавлении продуктов в таблице «Products» будет срабатывать хранимая процедура, которая добавляет блюдо в таблицу «Dish», добавляет значения в таблицу «Cooking» и обновляет значения в таблице «Products». Код хранимой процедуры представлен ниже.

CREATE OR REPLACE function three\_insert(name CHARACTER VARYING, name\_pr CHARACTER VARYING, amount\_pr INT) returns void

LANGUAGE plpgsql

AS $$

DECLARE

fk\_prod int;

fk\_dish int;

col\_pr int;

BEGIN

fk\_dish:= (SELECT id\_dish FROM Dish WHERE Name\_dish = name);

fk\_prod:= (SELECT Id\_product FROM Products WHERE Name\_product = name\_pr);

col\_pr:= (SELECT Count\_available FROM Products WHERE Name\_product = name\_pr);

-- if amount\_pr > col\_pr

-- THEN RAISE EXCEPTION 'Такого количества продукта нет на складе(';

-- END IF;

UPDATE Products set Count\_available = col\_pr - amount\_pr WHERE Name\_product = name\_pr;

INSERT INTO Cooking(Count\_of\_use\_products, fkey\_dish, fkey\_product) VALUES (amount\_pr, fk\_dish, fk\_prod);

END; $$;

При закупке продуктов в таблице «Purchase» будет срабатывать хранимая процедура, которая добавляет закупку в таблицу «Purchase» и обновляет значения в таблице «Products». Код хранимой процедуры представлен ниже.

CREATE OR REPLACE FUNCTION purchase\_add(name CHARACTER VARYING,col int) returns void

LANGUAGE plpgsql

AS $$

DECLARE

product\_id int;

count\_product int;

price int;

date date;

BEGIN

product\_id:=(SELECT Id\_product FROM Products WHERE Name\_product = name);

count\_product:=(SELECT Count\_available FROM Products WHERE Name\_product = name);

UPDATE Products set Count\_available = count\_product+col WHERE Name\_product = name;

price:= col\*10;

date:= current\_date;

INSERT INTO Purchase(Price\_purchase,Date\_of\_supply,Count\_of\_purchase\_products, fkey\_product) VALUES (price,date,col,product\_id);

END;

$$;

При загрузке страницы «Отчет о продуктах для приготовления» будет срабатывать функция, которая выводит суммарное количество используемых продуктов и использованное количество раз продукта для приготовления блюд. Код функции представлен ниже.

exports.getWork = async function(){

let resault;

await db.any('select \* from Purchase LEFT JOIN Products

ON fkey\_product = Id\_product where

date\_part(\'month\', Date\_of\_supply) = date\_part(\'month\', now())')

.then(function (data) {

for (let i = 0; i < data.length; i++){

data[i]['date\_of\_supply'] = transformationsDate(data[i]['date\_of\_supply']);

}

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = document.getElementById("new-tamplate").innerHTML;

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

}

При загрузке страницы «Работа ресторана/месяц» будет срабатывать функция, которая выводит все закупки продуктов за текущий месяц. Код функции представлен ниже.

exports.getWork = async function(){

let resault;

await db.any('select \* from Purchase LEFT JOIN Products ON

fkey\_product = Id\_product where date\_part(\'month\',

Date\_of\_supply) = date\_part(\'month\', now())')

.then(function (data) {

for (let i = 0; i < data.length; i++){

data[i]['date\_of\_supply'] = transformationsDate(data[i]['date\_of\_supply']);

}

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = document.getElementById("new-tamplate").innerHTML;

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

}

Полный код программы представлен в приложении А.

* 1. Проектирование справочной системы приложения

Для корректной работы с приложением требуется обеспечить пользователя справочной системой, в которой будут приведены приемы работы с приложением, включающие данные о том, что произойдёт после нажатия на определенную кнопку или при выборе вкладки.

Справочная система необходима для ознакомления с программой. В ней должна присутствовать информация, которая поможет в решении проблемы с приложением, а также может напомнить, как пользоваться программным средством.

Система справки данного приложения будет содержать следующие разделы:

* «Блюда»;
* «Продукты»;
* «Работа ресторана/месяц»;
* «Отчет о продуктах для приготовления»;
* «Закупка».

Справочная система будет создана отдельной кнопкой в навигационном меню на каждой странице приложения.

1. Описание программного средства
   1. Общие сведения

Приложение «Slavynie.exe» предназначено для учета закупки и затрат продуктов в ресторане. Само приложение весит 200 Мбайт. Для успешного использования программы на компьютере должно быть свободного места на диске не меньше 300 Мбайт, оперативная память должна быть от 1 Гбайт.

Для запуска приложения необходимо запустить ярлык «Slavynie.exe».

Компьютер должен работать под управлением операционной системы, начиная с Windows 7 и выше.

Инсталляция производится с установочного диска.

* 1. Функциональное назначение

Назначение программного средство «Slavynie» является автоматизация ведения склада продуктов, формирование закупки продуктов, списание продуктов для изготовления блюд. В данной программе классом решаемых задач являются:

* возможность добавлять, изменять данные в таблицы;
* возможность выводить данные таблицы на печать;
* возможность искать данные в таблице.
  1. Входные и выходные данные

Входными является данные, вводимые пользователем в таблицы базы данных: «Блюда», «Закупка». Кроме того, входными данными являются данные вводимые пользователем в поле поиска данных.

При добавлении новой информации в таблицу «Блюда» необходимо ввести следующие данные:

* название;
* цену;
* название продукта;
* количество продукта.

При добавлении новой информации в таблицу «Закупка» необходимо ввести следующие данные:

* название продукта;
* количество.

Выходная информация будет представлена в виде отчетов экспортируемых в файлы

MS Office.

Сформированный отчет, содержащий информацию о товарах, представлен на рисунке 6.

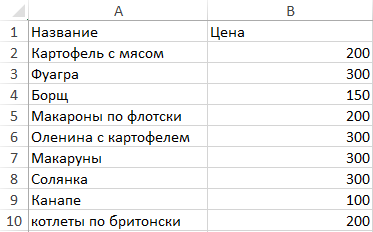


Рисунок 6 – Отчет о блюдах

Сформированный отчет, содержащий информацию о приготовлении продуктов, представлен на рисунке 7

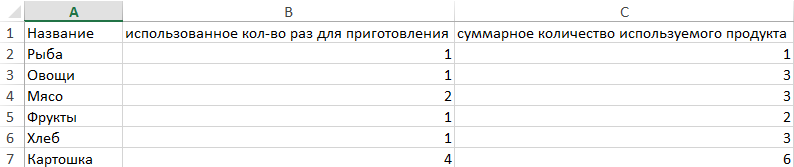


Рисунок 7 – Отчет о приготовлении продуктов

Сформированный отчет, содержащий информацию о закупках, представлен на рисунке 8.

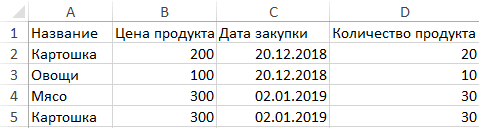


Рисунок 8 – Отчет о закупках

Сформированный отчет, содержащий информацию о продуктах, представлен на рисунке 9.



Рисунок 9. – Отчет о продуктах

1. Методика испытаний
   1. Технические требования

Минимальные системные требования к приложению:

* 300 Мбайт свободного места на винчестере;
* 1 Гбайт свободной оперативной памяти;
* процессор Intel Pentium 1.5 МГц или совместимый аналог;
* видеокарта с объемом памяти не менее 20 Мбайт.
  1. Функциональное тестирование

В процессе написания программного средства необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

Окно «Блюда» представлено на рисунке 10.

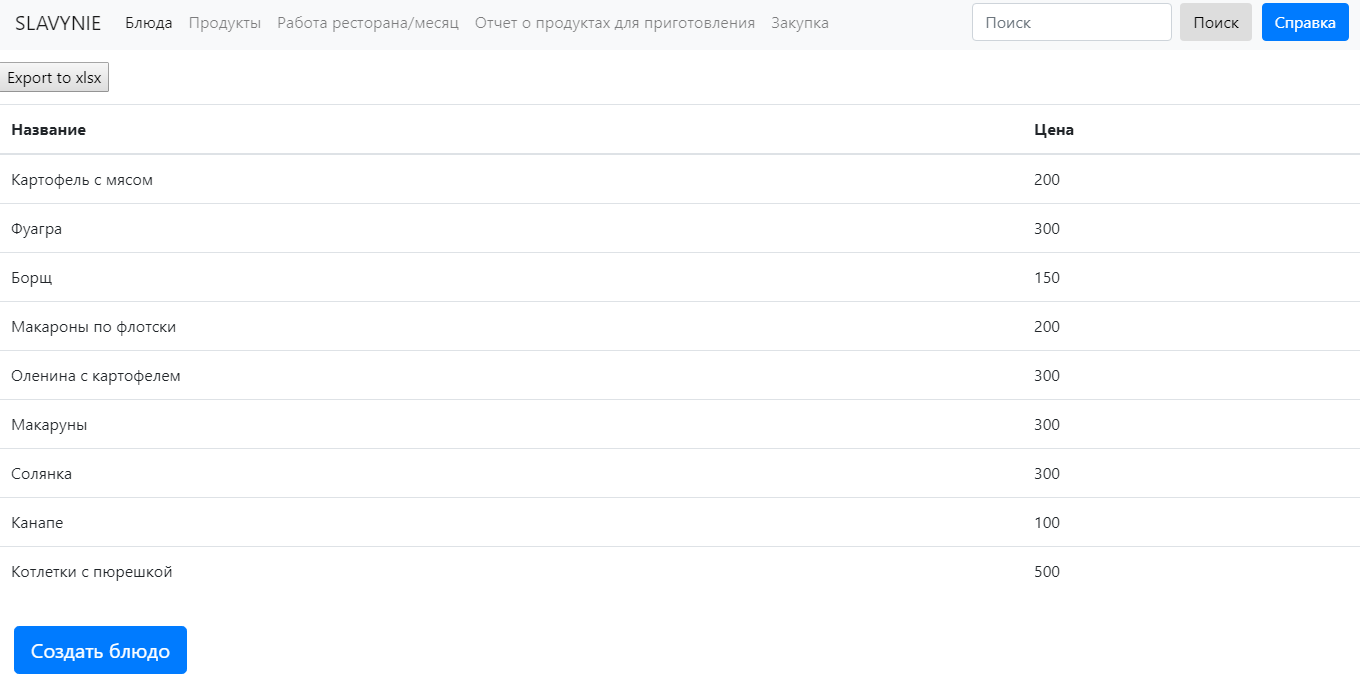


Рисунок 10 – Окно «Блюда»

Таблица 6 – Test-case для добавления блюда в базу данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 001 | Позитивный | Добавление данных о новом блюде. | 1. Нажать на кнопку «Создать блюдо». 2. Заполнить поля данными на форме «Создание блюда», представленной на рисунке 11:  * название: «котлеты по бритонски»; * цена: 200; * продукт: «мясо»; * количество продукта: 10; * нажать на кнопку «Добавить продукт»; * продукт: «овощи»; * количество продукта: 10.   3. Нажать кнопку «Отправить». | Ожидаемый результат: данные добавлены в таблицу «Блюда». |
| Фактический результат: результат соответствует ожидаемому. Все условия соблюдены. Результат добавления данных представлен на рисунке 12. |

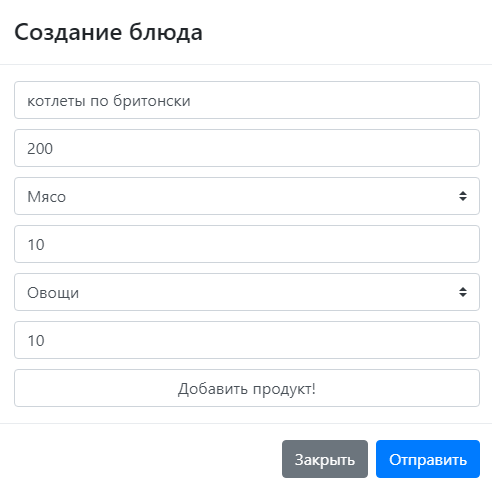


Рисунок 11 – Форма создания блюда

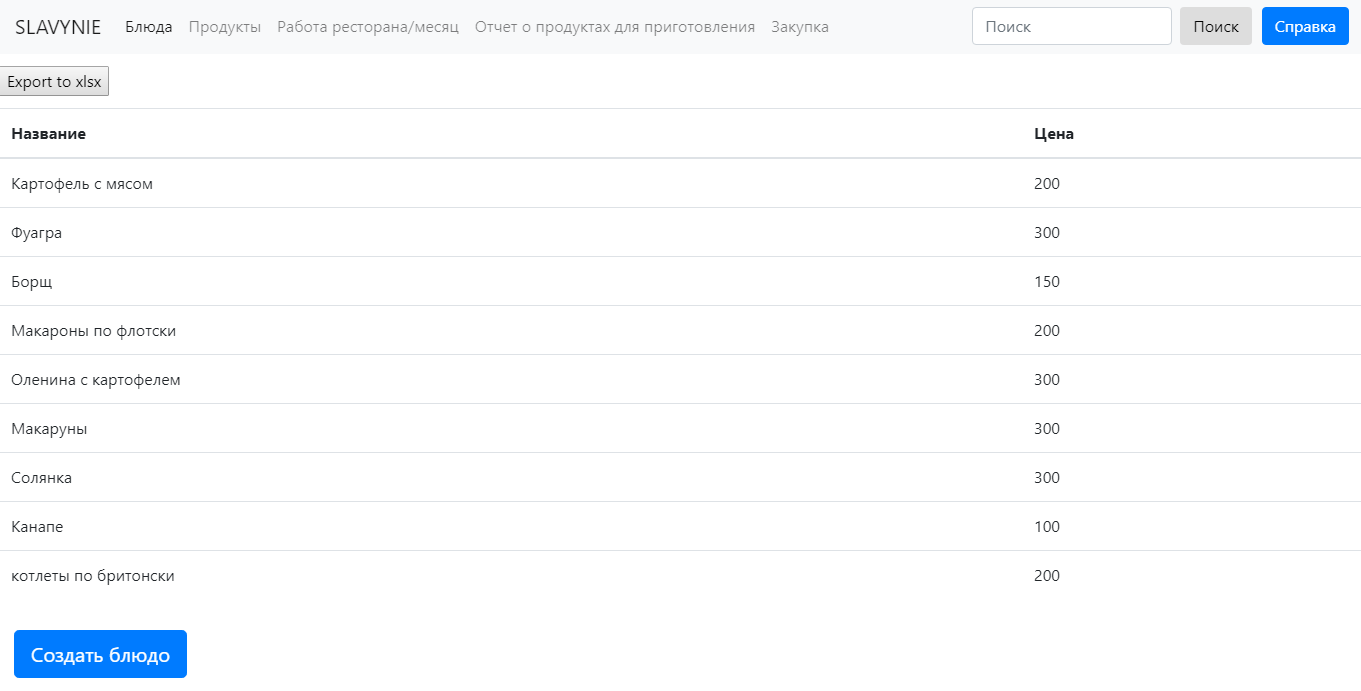


Рисунок 12 – Результат добавления блюда

Окно «Закупка» представлено на рисунке 13.

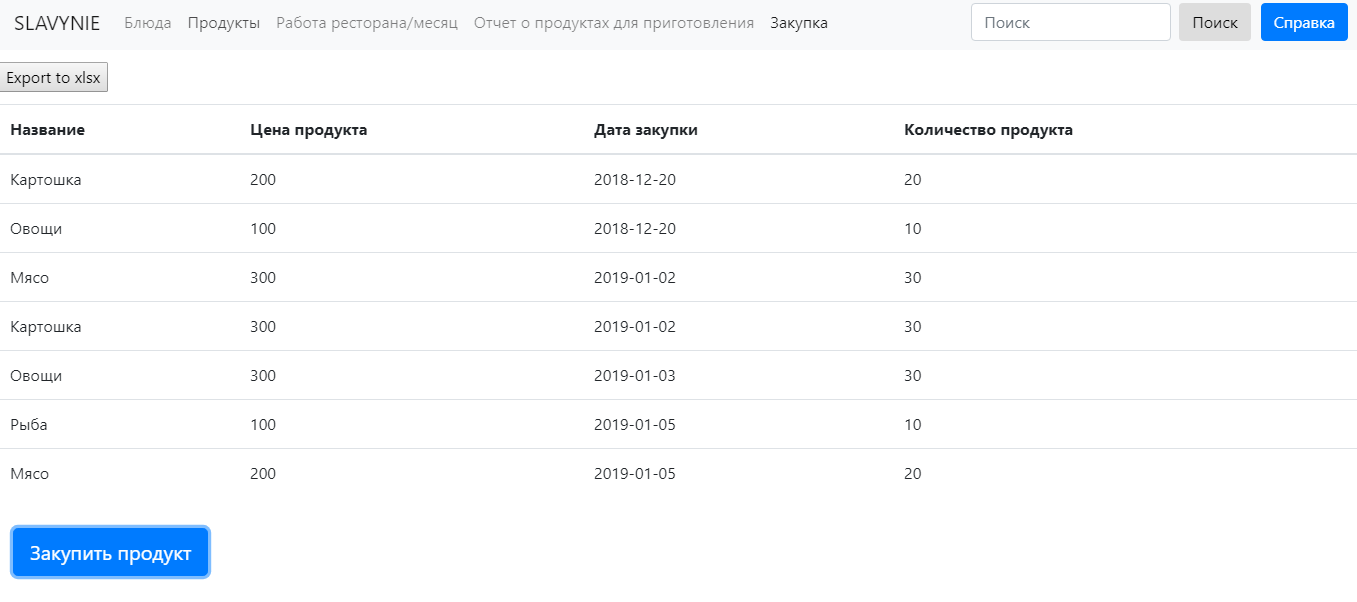


Рисунок 13 – Окно «Закупка»

Таблица 7 – Test-case для добавления закупки в базу данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 001 | Позитивный | Добавление данных о новой закупке. | 1. Нажать на кнопку «Закупить продукт». 2. Заполнить поля данными на форме «Закупка продукта», представленной на рисунке 14:  * продукт: «мясо»; * количество продукта: 50.   3. Нажать кнопку «Отправить». | Ожидаемый результат:  данные добавлены в таблицу «Закупки». |
| Фактический результат: результат соответствует ожидаемому. Все условия соблюдены. Результат добавления данных представлен на рисунке 15. |

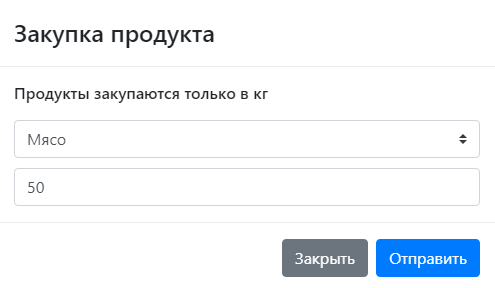


Рисунок 14 – Форма закупки продукта

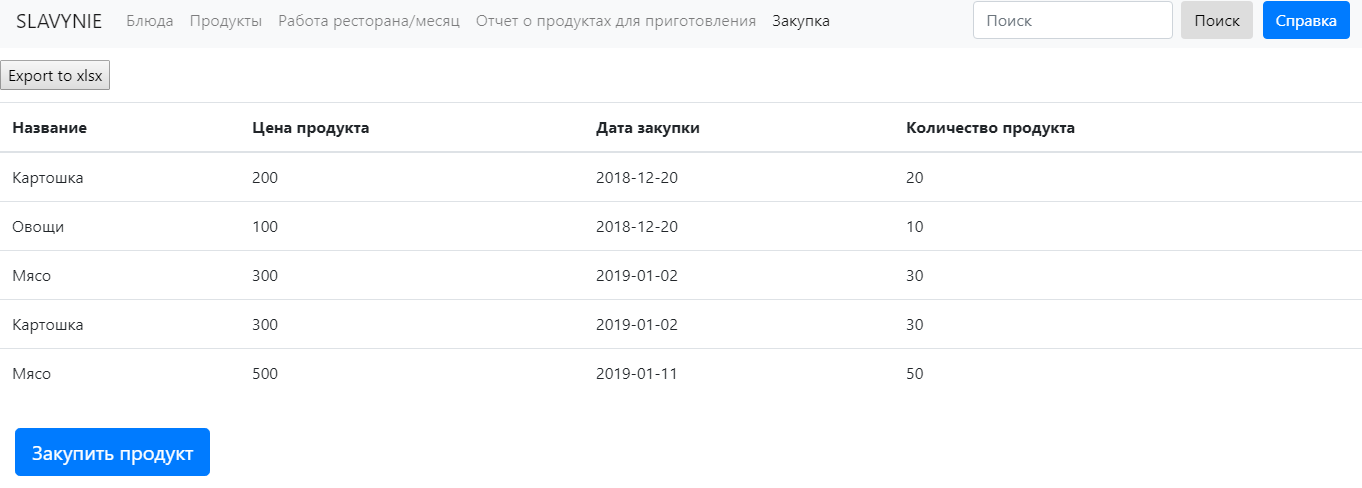


Рисунок 15 – Результат закупки продукта

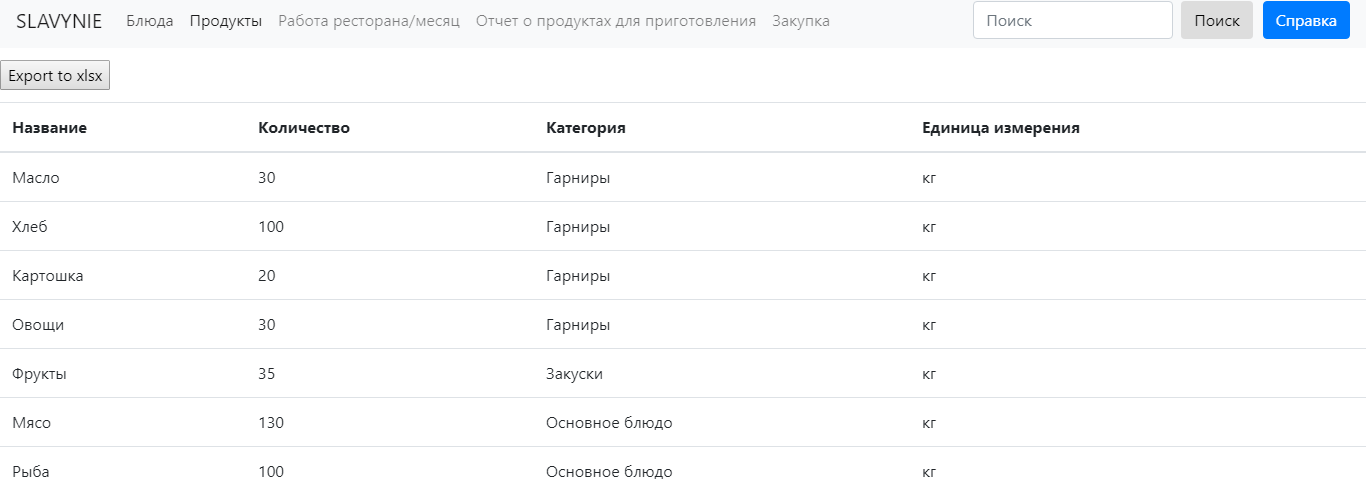


Рисунок 16 – Окно «Продукты»

Таблица 8 – Test-case для поиска данных в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 001 | Позитивный | Поиск данных в базе данных. | 1. Заполнить текстовое поле для поиска: «Мясо», представленного на рисунке 17. 2. Нажать на кнопку «Поиск». | Ожидаемый результат: при введении данных в поле для поиска найдётся введённое значение. |
| Фактический результат: результат соответствует ожидаемому. Все условия соблюдены. Результат поиска данных представлен на рисунке 18. |

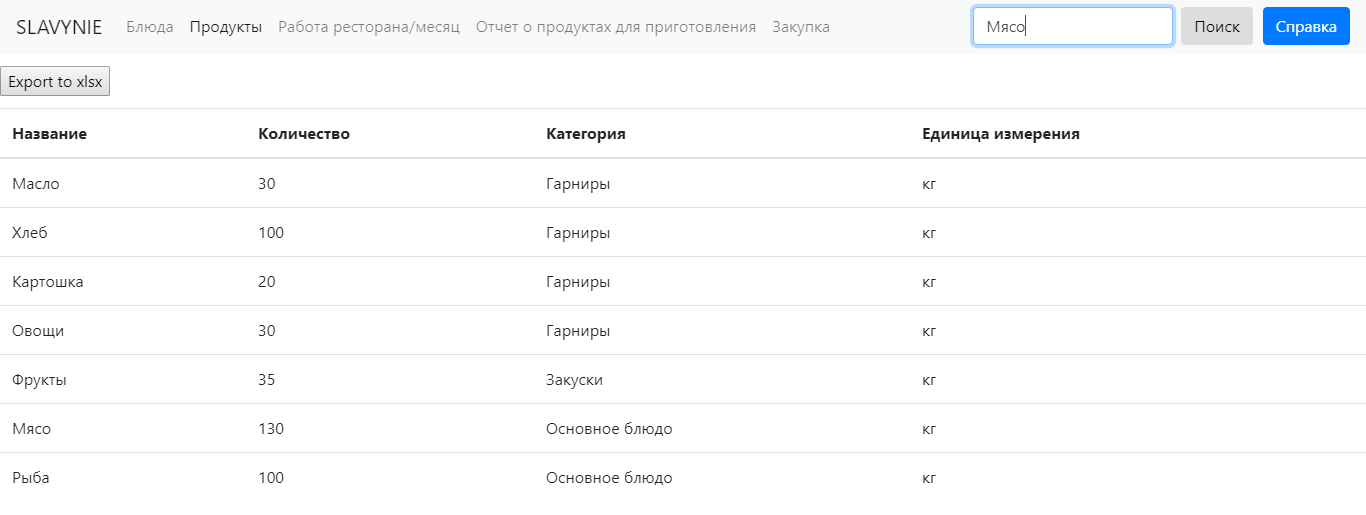


Рисунок 17 – Поле поиска продукта

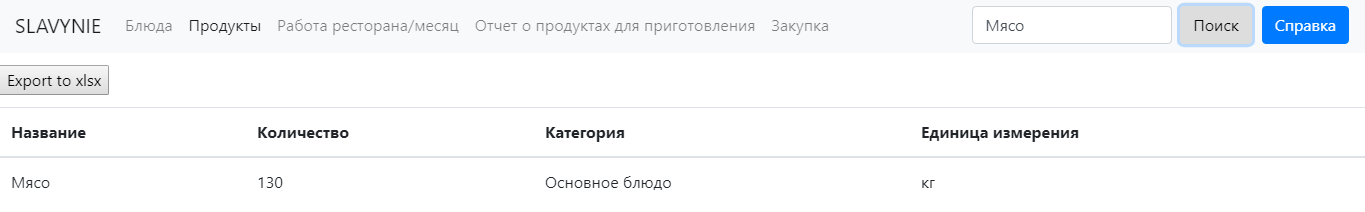


Рисунок 18 – Результат поиска продукта

Таблица 9 – Test-case для создания отчёта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 001 | Позитивный | Создание отчёта при нажатии на кнопку «Export to Exel». | 1. Нажать на кнопку «Продукты». 2. Нажать кнопку «Export to xlsx», представленную на рисунке 16; 3. Выбрать название и место сохранения файла, согласно рисунку 19. | Ожидаемый результат: при нажатии на кнопку создания отчёта и выбора места сохранения в выбранном месте появиться соответствующий файл. |
| Фактический результат: результат соответствует ожидаемому. Все условия соблюдены. Результат создания отчета представлена на рисунке 20. |

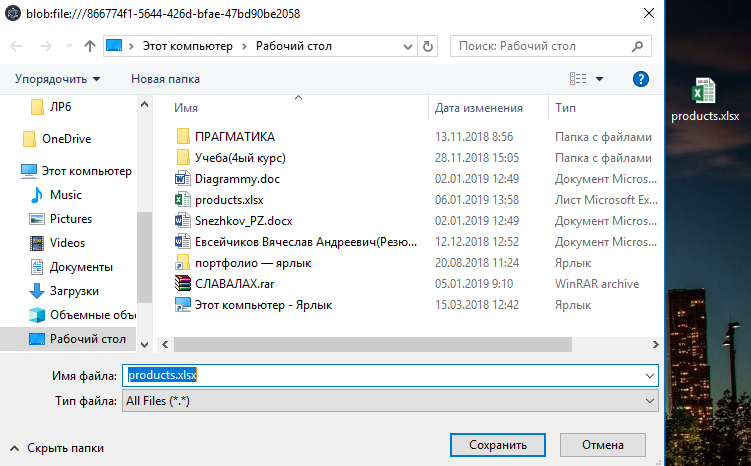


Рисунок 19 – Путь сохранения отчета



Рисунок 20 – Результат создания отчета

1. Применение
   1. Назначение программы

Приложение «Slavynie.exe» предназначено для автоматизации учета закупки и затрат продуктов в ресторане, который занимается изготовлением блюд. В этом приложении администратор ведёт учёт блюд, учет продуктов и их затраты для изготовления блюд, учет закупки продуктов, учет о закупке продуктов за текущий месяц, так же администратор может создать отчёт по необходимым таблицам базы данных и сохранения полученных отчётов в Microsoft Office.

* 1. Условия применения

Для применения данного программного средства необходимы следующие технически требования:

* процессор Intel Core 2 Duo или выше;
* минимальный объем оперативной памяти — 1 Гбайт;
* подключение к интернету, база данных располагается на сервере;
* операционная система windows 7 и выше;
* устройство для чтения дисков.
  1. Справочная система

Справочная система программного средства представляет собой модальное окно с полным описанием основных функций приложения. Модальное окно справки можно вызвать, нажав на кнопку «Справка», расположенную в навигационном меню приложения. В справочной системе даны ответы на типичные вопросы, возникающие при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Система справки данного приложения будет содержать следующие разделы:

* «Блюда»;
* «Продукты»;
* «Работа ресторана/месяц»;
* «Отчет о продуктах для приготовления»;
* «Закупка».

Справочная система будет создана отдельной кнопкой в навигационном меню на каждой странице приложения.

Содержимое справочной системы представлено на рисунке 21.

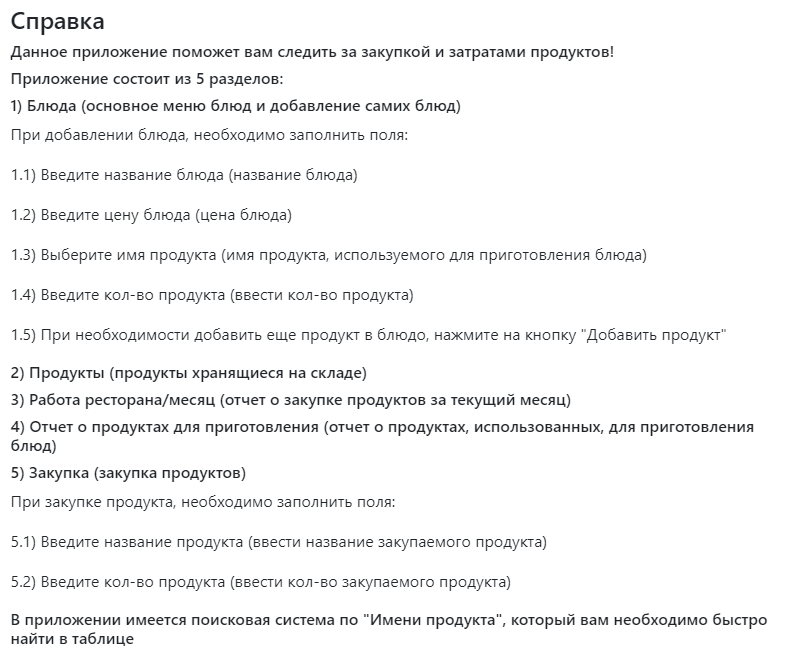


Рисунок 21 – Справочная система

Заключение

В рамках курсового проектирования по дисциплине «Базы данных и системы управления базами данных» было разработано приложение «Slavynie.exe», автоматизирующее учёт закупки и затрат продуктов в ресторане.

Для достижения цели курсового проектирования были решены следующие задачи:

* определена вычислительная система, необходимая для создания программного

средства;

* разработаны логическая и физическая модели данных;
* по модели выполнена проектирование задачи;
* разработано программное средство;
* описано созданное программное средство;
* выбрана методика испытания;
* описан процесс функционального тестирования;
* приведены примеры в области применения.

Разработка имеет интуитивно графический интерфейс, позволяющий с минимальным знанием компьютера использовать данное приложение.

В процессе разработке данного приложения были применены и закреплены знания по уже изученному материалу, были отработаны навыки владения методами надёжного программирования и эффективности разработки программного обеспечения в среде разработки текстового редактора с использованием языка программирования Java Script, разработана база данных средствами системы управления базами данных PostgreSQL.

Программа не реализована в полном объеме и не в полном соответствии с техническим заданием, однако отлажена и протестирована.

В следующих версияx программы планируется доделать сортировку данных таблиц по критериям.

Программа готова к практическому использованию в компании «Slavynie».

Список информационных источников

1. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск: КБП, 2017. – 30 c.
2. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г.Багласова, К.О.Якимович. – Минск: КБП, 2013. – 29 c.
3. Общие требования к тестовым документам: ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
4. Орлов, С. А. Технологии разработки программного обеспечения: Учебник для вузов. 4-е изд. / С. А. Орлов, Б. Я. Цилькер – СПб. : Питер, 2012. — 608 с.
5. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
6. Просницкий, А. Самоучитель «Управление проектами в Microsoft Project 2010» / А. Просницкий. – Київ, Казань, Красноярск, Рига, Санкт-Петербург, Львів, Чернівці, Івано-Франківськ, 2011.
7. Рудаков, А. В., Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования / А.В. Рудаков, Г.Н. Федорова М. : Издательский центр «Академия»; 2014. — 192 с.
8. Текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
9. [Atom](https://atom.io/) [Электронный ресурс] / Сайт программного средства. – atom.io, 2016-2018. – Режим доступа: https://www.atom.io. – Дата доступа: 01.10.2018.
10. coreldraw [Электронный ресурс] / Сайт программного средства. – coreldraw.com, 2006-2018. – Режим доступа: https://www.coreldraw.com/ru/. – Дата доступа: 10.10.2018.
11. metanit [Электронный ресурс] / Сайт о программировании. – metanit.com, 2012-2018. – Режим доступа: http://metanit.com. – Дата доступа: 14.10.2018.
12. postgresql [Электронный ресурс] / Сайт о программировании. – postgresql.org, 2005-2018. – Режим доступа: https://postgresql.org/docs – Дата доступа: 25.09.2018.
13. w3schools [Электронный ресурс] / Сайт о программировании. – w3schools.com, 1999-2018. – Режим доступа: https://www.w3schools.com/. – Дата доступа: 16.09.2018.

Приложение А

(обязательное)

Текст программных модулей

create schema Slavynie

Create table Products(

Id\_product serial PRIMARY KEY,

Name\_product text not null,

Count\_available int not null,

fkey\_category int not null,

Foreign key (fkey\_category) References Category\_of\_product (Id\_category)

);

Create table Category\_of\_product(

Id\_category serial PRIMARY KEY not null,

Name\_category text,

Unit\_of\_measurements varchar(4)

);

Create table Purchase(

Id\_purchase serial PRIMARY KEY not null,

Price\_purchase int not null,

Date\_of\_supply date not null,

Count\_of\_purchase\_products int not null,

fkey\_product int,

FOREIGN KEY (fkey\_product) REFERENCES Products (Id\_product)

);

Create table Cooking(

Id\_cooking serial PRIMARY KEY not null,

Count\_of\_use\_products int not null,

fkey\_dish int not null,

fkey\_product int not null,

FOREIGN KEY (fkey\_product) REFERENCES Products (Id\_product),

FOREIGN KEY (fkey\_dish) REFERENCES Dish (Id\_dish)

);

Create table Dish(

Id\_dish serial PRIMARY KEY not null,

Name\_dish text not null,

Price\_Dish int not null

);

//хранимая функция для добавления блюда

create or REPLACE function three\_insert(name CHARACTER VARYING,

name\_pr CHARACTER VARYING, amount\_pr INT) returns void

LANGUAGE plpgsql

AS $$

DECLARE

fk\_prod int;

fk\_dish int;

col\_pr int;

BEGIN

fk\_dish:= (SELECT id\_dish FROM Dish WHERE Name\_dish = name);

fk\_prod:= (SELECT Id\_product FROM Products WHERE Name\_product = name\_pr);

col\_pr:= (SELECT Count\_available FROM Products WHERE Name\_product = name\_pr);

UPDATE Products set Count\_available = col\_pr - amount\_pr WHERE Name\_product = name\_pr;

INSERT INTO Cooking(Count\_of\_use\_products, fkey\_dish, fkey\_product) VALUES

(amount\_pr, fk\_dish, fk\_prod);

END;

$$;

$(document).ready(function(){

$('#add\_dish').click(function(){

let name = String($('#name').val());

let price = String($('#price').val());

if(name == "" || price == "")

{

alert("Заполните все поля!");

}

else{

db.query('INSERT INTO Dish(name\_dish, price\_dish) VALUES($1,$2)',[name, price])

.then(async () => {

var formElements = [];

let tmpMas = document.getElementsByName("cik");

for(let i = 0; i < tmpMas.length; i++) {

formElements.push({

name: tmpMas[i].children[0].value,

amount: tmpMas[i].children[1].value

})

db.query("SELECT \* FROM three\_insert($1,$2,$3)",[name, formElements[i].name, formElements[i].amount])

}

location.reload();

})

.catch(error =>{

console.log(error);

})

}

});

$('#add\_product').click(function(){

let handle = $('#handle').html();

$('#clear\_div').append(handle);

})

});

//хранимая функция для добавления заказа

create or REPLACE function purchase\_add(name CHARACTER VARYING,col int) returns void

LANGUAGE plpgsql

AS $$

DECLARE

product\_id int;

count\_product int;

price int;

date date;

BEGIN

product\_id:=(SELECT Id\_product FROM Products WHERE Name\_product = name);

count\_product:=(SELECT Count\_available FROM Products WHERE Name\_product = name);

UPDATE Products set Count\_available = count\_product+col WHERE Name\_product = name;

price:= col\*10;

date:= current\_date;

INSERT INTO Purchase(Price\_purchase,Date\_of\_supply,Count\_of\_purchase\_products, fkey\_product)

VALUES (price,date,col,product\_id);

END;

$$;

$(document).ready(function(){

$('#add\_purchase').click(function(){

let name = String($('#name').val());

let col = String($('#col').val());

if(name == "" || col == "")

{

alert("Заполните все поля!");

}

else{

db.none('select \* from purchase\_add($1,$2)',[name, col])

.then(() => {

location.reload();

})

.catch(error =>{

console.log(error);

location.reload();

})

}

});

});

//функция поиска блюд

$(document).ready(function(){

$('#search\_purchase').click(async function(){

let search = String($('#purchase').val());

let resault;

await db.any('SELECT \* FROM Purchase LEFT JOIN Products ON Purchase.fkey\_product = Products.Id\_product

WHERE Products.name\_product = $1',[search])

.then(function (data) {

for (let i = 0; i < data.length; i++){

data[i]['date\_of\_supply'] = transformationsDate(data[i]['date\_of\_supply']);

}

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = '{{#each news}} <tr id="news"> <td>{{name\_product}}</td> <td>{{price\_purchase}}</td>

<td>{{date\_of\_supply}}</td> <td>{{count\_of\_purchase\_products}}</td>

</tr> {{/each}}';

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

});

});

//функция поиска продуктов

$(document).ready(function(){

$('#search\_product').click(async function(){

let search = String($('#product').val());

let resault;

await db.any('SELECT \* from Products left join Category\_of\_product ON

Products.fkey\_category = Category\_of\_product.Id\_category WHERE Products.name\_product = $1',[search])

.then(function (data) {

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = '{{#each news}}<tr id="news"><td>{{name\_product}}</td>

<td>{{count\_available}}</td><td>{{name\_category}}</td>

<td>{{unit\_of\_measurements}}</td></tr>{{/each}}';

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

});

});

//функция поиска блюд

$(document).ready(function(){

$('#search\_dish').click(async function(){

let search = String($('#dish').val());

let resault;

await db.any('SELECT \* from Dish WHERE name\_dish = $1',[search])

.then(function (data) {

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = '{{#each news}}<tr id="news"><td>{{name\_dish}}</td>

<td>{{price\_dish}}</td></tr>{{/each}}';

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

});

});

//функция поиска работы ресторана/месяц

$(document).ready(function(){

$('#search\_work').click(async function(){

let search = String($('#work').val());

let resault;

await db.any('select \* from Purchase LEFT JOIN Products ON fkey\_product = Id\_product

where date\_part(\'month\', Date\_of\_supply) = date\_part(\'month\', now())

AND Products.name\_product = $1',[search])

.then(function (data) {

for (let i = 0; i < data.length; i++){

data[i]['date\_of\_supply'] = transformationsDate(data[i]['date\_of\_supply']);

}

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = '{{#each news}}<tr id="news"><td>{{price\_purchase}}</td><td>{{date\_of\_supply}}</td>

<td>{{count\_of\_purchase\_products}}</td><td>{{name\_product}}</td></tr>{{/each}}';

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

});

});

//функция поиска отчета продуктов по приготовлению

$(document).ready(function(){

$('#search\_otchet').click(async function(){

let search = String($('#otchet').val());

let resault;

await db.any('SELECT name\_product, SUM(count\_of\_use\_products), count(Count\_of\_use\_products) FROM

Cooking LEFT JOIN Products ON fkey\_product = Id\_product

WHERE name\_product = $1 GROUP BY Name\_product',[search])

.then(function (data) {

resault = data;

сonsole.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = '{{#each news}}<tr id="news"><td scope="row">{{name\_product}}</td>

<td>{{count}}</td>

<td>{{sum}}</td></tr>{{/each}}';

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

});

});

//функция экспорта в Exel

function transformationsDate(date){

date = moment(date).format('YYYY-MM-DD');

return date;

};

let table = $('#table1').tableExport({

bootstrap: false,

formats: ['xlsx'],

filename: 'dish',

position: 'top',

});

//функция вывода данных о блюдах

exports.getDish = async function(){

let resault;

await db.any('SELECT \* from Dish')

.then(function (data) {

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = document.getElementById("new-tamplate").innerHTML;

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

}

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Название</th>

<th scope="col">Цена</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{name\_dish}}</td>

<td>{{price\_dish}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

//функция вывода продуктов для блюд в форму «Создать блюдо»

exports.getDishOption = async function(){

let resault;

await db.any('SELECT \* from Products')

.then(function (data) {

resault = data;

data.map((data, index)=>{

$("#name\_product").append($('<option id="count" value="'+ data.name\_product +'">'+

data.name\_product +'</option>'));

})

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

}

//функция вывода данных о продуктах

exports.getProduct = async function(){

let resault;

await db.any('SELECT \* from Products left join Category\_of\_product ON

Products.fkey\_category = Category\_of\_product.Id\_category')

.then(function (data) {

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = document.getElementById("new-tamplate").innerHTML;

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

}

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Название</th>

<th scope="col">Количество</th>

<th scope="col">Категория</th>

<th scope="col">Единица измерения</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{name\_product}}</td>

<td>{{count\_available}}</td>

<td>{{name\_category}}</td>

<td>{{unit\_of\_measurements}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

//функция вывода данных о закупках

exports.getPurchase = async function(){

let resault;

await db.any('SELECT \* FROM Purchase LEFT JOIN Products ON Purchase.fkey\_product = Products.Id\_product')

.then(function (data) {

for (let i = 0; i < data.length; i++){

data[i]['date\_of\_supply'] = transformationsDate(data[i]['date\_of\_supply']);

}

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = document.getElementById("new-tamplate").innerHTML;

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

}

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Название</th>

<th scope="col">Цена продукта</th>

<th scope="col">Дата закупки</th>

<th scope="col">Количество продукта</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{name\_product}}</td>

<td>{{price\_purchase}}</td>

<td>{{date\_of\_supply}}</td>

<td>{{count\_of\_purchase\_products}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

//функция вывода продуктов для закупки в форму «Создать закупку»

exports.getPurchaseOption = async function(){

let resault;

await db.any('SELECT \* from Products')

.then(function (data) {

resault = data;

data.map((data, index)=>{

$("#name").append($('<option id="count" value="'+ data.name\_product +'">'+ data.name\_product +'</option>'));

})

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

}

//функция вывода данных о работе ресторана/месяц

exports.getWork = async function(){

let resault;

await db.any('select \* from Purchase LEFT JOIN Products ON

fkey\_product = Id\_product where date\_part(\'month\', Date\_of\_supply) = date\_part(\'month\', now())')

.then(function (data) {

for (let i = 0; i < data.length; i++){

data[i]['date\_of\_supply'] = transformationsDate(data[i]['date\_of\_supply']);

}

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = document.getElementById("new-tamplate").innerHTML;

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

.catch(function (error) {

console.error('ERROR:', error);

})

return resault;

}

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Цена</th>

<th scope="col">Дата</th>

<th scope="col">Количество</th>

<th scope="col">Название</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{price\_purchase}}</td>

<td>{{date\_of\_supply}}</td>

<td>{{count\_of\_purchase\_products}}</td>

<td>{{name\_product}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

//функия вывода данных об отчете продуктов за приготовление блюд

exports.getOtchet = async function(){

let resault;

await db.any('SELECT name\_product, SUM(count\_of\_use\_products), count(Count\_of\_use\_products)

FROM Cooking LEFT JOIN Products ON fkey\_product = Id\_product GROUP BY Name\_product')

.then(function (data) {

resault = data;

console.log(data);

var json = JSON.parse(JSON.stringify({news: data}));

var rawTamplate = document.getElementById("new-tamplate").innerHTML;

var compiledTamplate = Handlebars.compile(rawTamplate);

var genHTML = compiledTamplate(json);

var news = document.getElementById("container");

news.innerHTML = genHTML;

})

}

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Название</th>

<th scope="col">использованное кол-во раз для приготовления</th>

<th scope="col">суммарное количество используемого продукта</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{name\_product}}</td>

<td>{{count}}</td>

<td>{{sum}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" integrity="sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB" crossorigin="anonymous">

<script>

require("bootstrap");

var $ = require("jquery");

require('./sel.js').getDishOption();

require('./sel.js').getDish();

require('./addDish.js');

require('./search.js');

</script>

<title>Программа по учету закупки и затрат продуктов</title>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="dish.html">SLAVYNIE</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNavAltMarkup">

<div class="navbar-nav">

<a class="nav-item nav-link active" href="dish.html">Блюда <span class="sr-only">(current)</span></a>

<a class="nav-item nav-link" href="product.html">Продукты</a>

<a class="nav-item nav-link" href="work\_of\_restaurant.html">Работа ресторана/месяц</a>

<a class="nav-item nav-link" href="otchet.html">Отчет о продуктах для приготовления</a>

<a class="nav-item nav-link" href="purchase.html">Закупка</a>

</div>

<form class="form-inline ml-auto">

<div class="md-form my-0">

<input class="form-control" id="dish" type="text" placeholder="Поиск" aria-label="Search">

</div>

<input id="search\_dish" class="btn btn-outline-white btn-md my-0 ml-sm-2" value="Поиск" type="button">

</form>

<button style="margin-left: 10px;" class="btn btn-primary" data-toggle="modal" data-target=".bd-example-modal-lg">Справка</button>

<div class="modal fade bd-example-modal-lg" tabindex="-1" role="dialog" aria-labelledby="myLargeModalLabel" aria-hidden="true">

<div class="modal-dialog modal-lg">

<div class="modal-content">

<div class="modal-body">

<h4>Справка</h4>

<h6>Данное приложение поможет вам следить за закупкой и затратами продуктов!</h6>

<h6>Приложение состоит из 5 разделов:</h6>

<h6>1) Блюда (основное меню блюд и добавление самих блюд)</h6>

<p>При добавлении блюда, необходимо заполнить поля:</p>

<p>1.1) Введите название блюда (название блюда)</p>

<p>1.2) Введите цену блюда (цена блюда)</p>

<p>1.3) Выберите имя продукта (имя продукта, используемого для приготовления блюда)</p>

<p>1.4) Введите кол-во продукта (ввести кол-во продукта)</p>

<p>1.5) При необходимости добавить еще продукт в блюдо, нажмите на кнопку "Добавить продукт"</p>

<h6>2) Продукты (продукты хранящиеся на складе)</h6>

<h6>3) Работа ресторана/месяц (отчет о закупке продуктов за текущий месяц)</h6>

<h6>4) Отчет о продуктах для приготовления (отчет о продуктах, использованных, для приготовления блюд)</h6>

<h6>5) Закупка (закупка продуктов)</h6>

<p>При закупке продукта, необходимо заполнить поля:</p>

<p>5.1) Введите название продукта (ввести название закупаемого продукта)</p>

<p>5.2) Введите кол-во продукта (ввести кол-во закупаемого продукта)</p>

<h6>В приложении имеется поисковая система по "Имени продукта", который вам необходимо быстро найти в таблице</h6>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

</div>

</div>

</div>

</div>

</div>

</nav>

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Название</th>

<th scope="col">Цена</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{name\_dish}}</td>

<td>{{price\_dish}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

<button style="margin-left: 15px; margin-top: 15px" type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">

Создать блюдо

</button>

<!-- Модальное окно -->

<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title" id="myModalLabel">Создание блюда</h4>

</div>

<div class="modal-body">

<input class="form-control" type="text" placeholder="Введите название" id="name">

<input style="margin-top: 10px" class="form-control" type="text" placeholder="Введите цену" id="price">

<!-- <input style="margin-top: 10px" class="form-control" type="text" placeholder="Выбрать имя продукта 1" id="name\_product"> -->

<div name="hand" id="handle">

<div name="cik" id="cikl">

<select name="n\_prod" style="margin-top: 10px" class="custom-select" id="name\_product">

<option value="">Выберите продукт</option>

</select>

<input style="margin-top: 10px" name="c\_prod" class="form-control" type="text" placeholder="Введите кол-во продукта" id="col\_product">

</div>

</div>

<div id="clear\_div"></div>

<input style="margin-top: 10px;" type="button" class="form-control" value="Добавить продукт!" id="add\_product">

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

<button type="button" id="add\_dish" class="btn btn-primary">Отправить</button>

</div>

</div>

</div>

</div>

<script>

require('./renderer.js')

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" integrity="sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB" crossorigin="anonymous">

<script>

require("bootstrap");

require('./sel.js').getProduct();

require('./search.js');

</script>

<title>Программа по учету закупки и затрат продуктов</title>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="dish.html">SLAVYNIE</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNavAltMarkup">

<div class="navbar-nav">

<a class="nav-item nav-link" href="dish.html">Блюда <span class="sr-only">(current)</span></a>

<a class="nav-item nav-link active" href="product.html">Продукты</a>

<a class="nav-item nav-link" href="work\_of\_restaurant.html">Работа ресторана/месяц</a>

<a class="nav-item nav-link" href="otchet.html">Отчет о продуктах для приготовления</a>

<a class="nav-item nav-link" href="purchase.html">Закупка</a>

</div>

<form class="form-inline ml-auto">

<div class="md-form my-0">

<input class="form-control" id="product" type="text" placeholder="Поиск" aria-label="Search">

</div>

<input id="search\_product" class="btn btn-outline-white btn-md my-0 ml-sm-2" value="Поиск" type="button">

</form>

<button style="margin-left: 10px;" class="btn btn-primary" data-toggle="modal" data-target=".bd-example-modal-lg">Справка</button>

<div class="modal fade bd-example-modal-lg" tabindex="-1" role="dialog" aria-labelledby="myLargeModalLabel" aria-hidden="true">

<div class="modal-dialog modal-lg">

<div class="modal-content">

<div class="modal-body">

<h4>Справка</h4>

<h6>Данное приложение поможет вам следить за закупкой и затратами продуктов!</h6>

<h6>Приложение состоит из 5 разделов:</h6>

<h6>1) Блюда (основное меню блюд и добавление самих блюд)</h6>

<p>При добавлении блюда, необходимо заполнить поля:</p>

<p>1.1) Введите название блюда (название блюда)</p>

<p>1.2) Введите цену блюда (цена блюда)</p>

<p>1.3) Выберите имя продукта (имя продукта, используемого для приготовления блюда)</p>

<p>1.4) Введите кол-во продукта (ввести кол-во продукта)</p>

<p>1.5) При необходимости добавить еще продукт в блюдо, нажмите на кнопку "Добавить продукт"</p>

<h6>2) Продукты (продукты хранящиеся на складе)</h6>

<h6>3) Работа ресторана/месяц (отчет о закупке продуктов за текущий месяц)</h6>

<h6>4) Отчет о продуктах для приготовления (отчет о продуктах, использованных, для приготовления блюд)</h6>

<h6>5) Закупка (закупка продуктов)</h6>

<p>При закупке продукта, необходимо заполнить поля:</p>

<p>5.1) Введите название продукта (ввести название закупаемого продукта)</p>

<p>5.2) Введите кол-во продукта (ввести кол-во закупаемого продукта)</p>

<h6>В приложении имеется поисковая система по "Имени продукта", который вам необходимо быстро найти в таблице</h6>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

</div>

</div>

</div>

</div>

</div>

</nav>

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Название</th>

<th scope="col">Количество</th>

<th scope="col">Категория</th>

<th scope="col">Единица измерения</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{name\_product}}</td>

<td>{{count\_available}}</td>

<td>{{name\_category}}</td>

<td>{{unit\_of\_measurements}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

<script>

// You can also require other files to run in this process

require('./renderer.js')

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" integrity="sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB" crossorigin="anonymous">

<script>

require("bootstrap");

require('./sel.js').getWork();

require('./search.js');

</script>

<title>Программа по учету закупки и затрат продуктов</title>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="dish.html">SLAVYNIE</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNavAltMarkup">

<div class="navbar-nav">

<a class="nav-item nav-link" href="dish.html">Блюда <span class="sr-only">(current)</span></a>

<a class="nav-item nav-link" href="product.html">Продукты</a>

<a class="nav-item nav-link active" href="work\_of\_restaurant.html">Работа ресторана/месяц</a>

<a class="nav-item nav-link" href="otchet.html">Отчет о продуктах для приготовления</a>

<a class="nav-item nav-link" href="purchase.html">Закупка</a>

</div>

<form class="form-inline ml-auto">

<div class="md-form my-0">

<input class="form-control" id="work" type="text" placeholder="Поиск" aria-label="Search">

</div>

<input id="search\_work" class="btn btn-outline-white btn-md my-0 ml-sm-2" value="Поиск" type="button">

</form>

<button style="margin-left: 10px;" class="btn btn-primary" data-toggle="modal" data-target=".bd-example-modal-lg">Справка</button>

<div class="modal fade bd-example-modal-lg" tabindex="-1" role="dialog" aria-labelledby="myLargeModalLabel" aria-hidden="true">

<div class="modal-dialog modal-lg">

<div class="modal-content">

<div class="modal-body">

<h4>Справка</h4>

<h6>Данное приложение поможет вам следить за закупкой и затратами продуктов!</h6>

<h6>Приложение состоит из 5 разделов:</h6>

<h6>1) Блюда (основное меню блюд и добавление самих блюд)</h6>

<p>При добавлении блюда, необходимо заполнить поля:</p>

<p>1.1) Введите название блюда (название блюда)</p>

<p>1.2) Введите цену блюда (цена блюда)</p>

<p>1.3) Выберите имя продукта (имя продукта, используемого для приготовления блюда)</p>

<p>1.4) Введите кол-во продукта (ввести кол-во продукта)</p>

<p>1.5) При необходимости добавить еще продукт в блюдо, нажмите на кнопку "Добавить продукт"</p>

<h6>2) Продукты (продукты хранящиеся на складе)</h6>

<h6>3) Работа ресторана/месяц (отчет о закупке продуктов за текущий месяц)</h6>

<h6>4) Отчет о продуктах для приготовления (отчет о продуктах, использованных, для приготовления блюд)</h6>

<h6>5) Закупка (закупка продуктов)</h6>

<p>При закупке продукта, необходимо заполнить поля:</p>

<p>5.1) Введите название продукта (ввести название закупаемого продукта)</p>

<p>5.2) Введите кол-во продукта (ввести кол-во закупаемого продукта)</p>

<h6>В приложении имеется поисковая система по "Имени продукта", который вам необходимо быстро найти в таблице</h6>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

</div>

</div>

</div>

</div>

</div>

</nav>

<h5 style="margin-left: 15px; margin-top: 15px;">Отчет закупки продуктов за текущий месяц</h5>

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Цена</th>

<th scope="col">Дата</th>

<th scope="col">Количество</th>

<th scope="col">Название</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{price\_purchase}}</td>

<td>{{date\_of\_supply}}</td>

<td>{{count\_of\_purchase\_products}}</td>

<td>{{name\_product}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

<script>

// You can also require other files to run in this process

require('./renderer.js')

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" integrity="sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB" crossorigin="anonymous">

<script>

require("bootstrap");

var $ = require("jquery");

require('./sel.js').getOtchet();

require('./search.js');

</script>

<title>Программа по учету закупки и затрат продуктов</title>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="dish.html">SLAVYNIE</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNavAltMarkup">

<div class="navbar-nav">

<a class="nav-item nav-link" href="dish.html">Блюда <span class="sr-only">(current)</span></a>

<a class="nav-item nav-link" href="product.html">Продукты</a>

<a class="nav-item nav-link" href="work\_of\_restaurant.html">Работа ресторана/месяц</a>

<a class="nav-item nav-link active" href="otchet.html">Отчет о продуктах для приготовления</a>

<a class="nav-item nav-link" href="purchase.html">Закупка</a>

</div>

<form class="form-inline ml-auto">

<div class="md-form my-0">

<input class="form-control" id="otchet" type="text" placeholder="Поиск" aria-label="Search">

</div>

<input id="search\_otchet" class="btn btn-outline-white btn-md my-0 ml-sm-2" value="Поиск" type="button">

</form>

<button style="margin-left: 10px;" class="btn btn-primary" data-toggle="modal" data-target=".bd-example-modal-lg">Справка</button>

<div class="modal fade bd-example-modal-lg" tabindex="-1" role="dialog" aria-labelledby="myLargeModalLabel" aria-hidden="true">

<div class="modal-dialog modal-lg">

<div class="modal-content">

<div class="modal-body">

<h4>Справка</h4>

<h6>Данное приложение поможет вам следить за закупкой и затратами продуктов!</h6>

<h6>Приложение состоит из 5 разделов:</h6>

<h6>1) Блюда (основное меню блюд и добавление самих блюд)</h6>

<p>При добавлении блюда, необходимо заполнить поля:</p>

<p>1.1) Введите название блюда (название блюда)</p>

<p>1.2) Введите цену блюда (цена блюда)</p>

<p>1.3) Выберите имя продукта (имя продукта, используемого для приготовления блюда)</p>

<p>1.4) Введите кол-во продукта (ввести кол-во продукта)</p>

<p>1.5) При необходимости добавить еще продукт в блюдо, нажмите на кнопку "Добавить продукт"</p>

<h6>2) Продукты (продукты хранящиеся на складе)</h6>

<h6>3) Работа ресторана/месяц (отчет о закупке продуктов за текущий месяц)</h6>

<h6>4) Отчет о продуктах для приготовления (отчет о продуктах, использованных, для приготовления блюд)</h6>

<h6>5) Закупка (закупка продуктов)</h6>

<p>При закупке продукта, необходимо заполнить поля:</p>

<p>5.1) Введите название продукта (ввести название закупаемого продукта)</p>

<p>5.2) Введите кол-во продукта (ввести кол-во закупаемого продукта)</p>

<h6>В приложении имеется поисковая система по "Имени продукта", который вам необходимо быстро найти в таблице</h6>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

</div>

</div>

</div>

</div>

</div>

</nav>

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Название</th>

<th scope="col">использованное кол-во раз для приготовления</th>

<th scope="col">суммарное количество используемого продукта</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{name\_product}}</td>

<td>{{count}}</td>

<td>{{sum}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

<script>

require('./renderer.js')

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" integrity="sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB" crossorigin="anonymous">

<script>

// var bd= require("./select.js");

require("bootstrap");

require('./sel.js').getPurchase();

require('./addPurchase.js');

require('./search.js');

require('./sel.js').getPurchaseOption();

</script>

<title>Программа по учету закупки и затрат продуктов</title>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="dish.html">SLAVYNIE</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNavAltMarkup">

<div class="navbar-nav">

<a class="nav-item nav-link" href="dish.html">Блюда <span class="sr-only">(current)</span></a>

<a class="nav-item nav-link" href="product.html">Продукты</a>

<a class="nav-item nav-link" href="work\_of\_restaurant.html">Работа ресторана/месяц</a>

<a class="nav-item nav-link" href="otchet.html">Отчет о продуктах для приготовления</a>

<a class="nav-item nav-link active" href="purchase.html">Закупка</a>

</div>

<form class="form-inline ml-auto">

<div class="md-form my-0">

<input class="form-control" id="purchase" type="text" placeholder="Поиск" aria-label="Search">

</div>

<input id="search\_purchase" class="btn btn-outline-white btn-md my-0 ml-sm-2" value="Поиск" type="button">

</form>

<button style="margin-left: 10px;" class="btn btn-primary" data-toggle="modal" data-target=".bd-example-modal-lg">Справка</button>

<div class="modal fade bd-example-modal-lg" tabindex="-1" role="dialog" aria-labelledby="myLargeModalLabel" aria-hidden="true">

<div class="modal-dialog modal-lg">

<div class="modal-content">

<div class="modal-body">

<h4>Справка</h4>

<h6>Данное приложение поможет вам следить за закупкой и затратами продуктов!</h6>

<h6>Приложение состоит из 5 разделов:</h6>

<h6>1) Блюда (основное меню блюд и добавление самих блюд)</h6>

<p>При добавлении блюда, необходимо заполнить поля:</p>

<p>1.1) Введите название блюда (название блюда)</p>

<p>1.2) Введите цену блюда (цена блюда)</p>

<p>1.3) Выберите имя продукта (имя продукта, используемого для приготовления блюда)</p>

<p>1.4) Введите кол-во продукта (ввести кол-во продукта)</p>

<p>1.5) При необходимости добавить еще продукт в блюдо, нажмите на кнопку "Добавить продукт"</p>

<h6>2) Продукты (продукты хранящиеся на складе)</h6>

<h6>3) Работа ресторана/месяц (отчет о закупке продуктов за текущий месяц)</h6>

<h6>4) Отчет о продуктах для приготовления (отчет о продуктах, использованных, для приготовления блюд)</h6>

<h6>5) Закупка (закупка продуктов)</h6>

<p>При закупке продукта, необходимо заполнить поля:</p>

<p>5.1) Введите название продукта (ввести название закупаемого продукта)</p>

<p>5.2) Введите кол-во продукта (ввести кол-во закупаемого продукта)</p>

<h6>В приложении имеется поисковая система по "Имени продукта", который вам необходимо быстро найти в таблице</h6>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

</div>

</div>

</div>

</div>

</div>

</nav>

<!-- <table style="margin-left: 15px; margin-top: 15px" border="1">

<tr>

<th style="width: 25px; text-align: center;">ID</th>

<th style="width: 200px; text-align: center;">Название</th>

<th style="width: 50px; text-align: center;">Цена продукта</th>

<th style="width: 150px; text-align: center;">Дата закупки</th>

<th style="width: 50px; text-align: center;">Кол-во продукта</th>

</tr>

</table>

<div id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<div id="news">

<table style="margin-left: 15px" border="1">

<tr><td style="width: 25px; text-align: center;">{{id\_purchase}}</td><td style="width: 200px; padding-left: 5px">{{name\_product}}</td><td style="width: 76px; text-align: center;">{{price\_purchase}}</td><td style="width: 150px; text-align: center;">{{date\_of\_supply}}</td><td style="width: 76px; text-align: center;">{{count\_of\_purchase\_products}}</td></tr>

</table>

</div>

{{/each}}

</script>

</div> -->

<table id="table1" class="table">

<thead>

<tr>

<th scope="col">Название</th>

<th scope="col">Цена продукта</th>

<th scope="col">Дата закупки</th>

<th scope="col">Количество продукта</th>

</tr>

</thead>

<tbody id="container">

<script id="new-tamplate" type="text/x-handlebars-template">

{{#each news}}

<tr id="news">

<td>{{name\_product}}</td>

<td>{{price\_purchase}}</td>

<td>{{date\_of\_supply}}</td>

<td>{{count\_of\_purchase\_products}}</td>

</tr>

{{/each}}

</script>

</tbody>

</table>

<button style="margin-left: 15px; margin-top: 15px" type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">

Закупить продукт

</button>

<!-- Модальное окно -->

<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h4 class="modal-title" id="myModalLabel">Закупка продукта</h4>

</div>

<div class="modal-body">

<h6>Продукты закупаются только в кг</h6>

<!-- <input class="form-control" type="text" placeholder="Введите название продукта 1" id="name"> -->

<select style="margin-top: 10px" class="custom-select" id="name">

<option value="">Выберите продукт</option>

</select>

<input style="margin-top: 10px" class="form-control" type="text" placeholder="Введите кол-во продукта 1" id="col">

<!-- <input style="margin-top: 10px" class="form-control" type="text" placeholder="Введите категорию продукта" id="kat">

<input style="margin-top: 10px" class="form-control" type="text" placeholder="Введите единицу измерения продукта" id="unit\_kat"> -->

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

<button type="button" id="add\_purchase" class="btn btn-primary">Отправить</button>

</div>

</div>

</div>

</div>

<!-- <table>

<td>

<tr class="name">{{name}}</tr>

<tr class="price">{{price}}</tr>

</td>

</table> -->

<script>

// You can also require other files to run in this process

require('./renderer.js')

</script>

</body>

</html>