**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

СОГЛАСОВАНО  
Кандидат технических наук, доцент департамента программной инженерии факультета компьютерных наук

\_\_\_\_\_\_\_\_\_\_\_\_\_ С. Л. Макаров  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

УТВЕРЖДАЮ  
Академический руководитель образовательной программы «Программная инженерия»  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**IOS-ПРИЛОЖЕНИЕ ДЛЯ ЗАКАЗА ЕДЫ**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.11-01 81 01-1-ЛУ**

**Исполнитель**

Студент группы БПИ173

\_\_\_\_\_\_\_\_\_\_\_\_ / Д. П. Королев /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**Москва 2018**

**УТВЕРЖДЕН**

**RU.17701729.04.11-01 81 01-1-ЛУ**

**IOS-ПРИЛОЖЕНИЕ ДЛЯ ЗАКАЗА ЕДЫ**

**Пояснительная записка**

**RU.17701729.04.11-01 81 01-1**

**Листов 34**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**Москва 2018**

**СОДЕРЖАНИЕ**

[1. ВВЕДЕНИЕ](#Введение) 3

[1.1. Наименование программы](#Наименованиепрограммы) 3

[1.2. Документы на основании которых ведётся разработка](#Документынаоснованиикоторыхведется) 3

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ](#НАЗНАЧЕНИЕИОБЛАСТЬПРИМЕНЕНИЯ) 4

[2.1. Функциональное назначение](#Функциональноеназначение) 4

[2.2. Эксплуатационное назначение](#Эксплуатационноеназначение) 4

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ](#ТЕХНИЧЕСКИЕХАРАКТЕРИСТИКИ) 5

[3.1. Постановка задачи на разработку программы](#Постановказадачинаразработкупрограмм) 5

[3.2. Описание функционирования программы](#Описаниефункционированияпрограммы) 5

[3.2.1 Алгоритм загрузки продуктов с сервера](#Алгоритмзагрузкипродуктовссервера) 6

[3.2.2 Реализация серверной части приложения](#СЕРВЕР) 7

[3.2.3 Реализация клиентской части приложения](#КЛИЕНТ) 9

[3.3 Описание и обоснование метода организации входных и выходных данных](#Описаниеиобоснованиеметодаорганизаци) 10

[3.4 Описание и обоснование выбора технических и программных средств. 1](#Описаниеиобоснованиевыборатехнически)1

[3.5 Состав рекомендуемых технических и программных средств 1](#СОСТАВ)1

[4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 1](#ОЖИДАЕМЫЕТЕХНИКОЭКОНОМИЧЕСКИЕПОКАЗ)3

[4.1. Ориентировочная экономическая эффективность 1](#Ориентировочнаяэкономическаяэффективно)3

[4.2. Предполагаемая потребность 1](#Предполагаемаяпотребность)3

[4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами 1](#Экономическиепреимуществаразработкипо)3

[5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 1](#ИСТОЧНИКИИСПОЛЬЗОВАННЫЕПРИРАЗРАБО)4

[Приложение 1 «Описание функциональности классов сервера\клиента» 15](#Приложение1)

[Приложение 2 «Описание полей, свойств и методов классов сервера\клиента» 16](#Приложение2)

[Приложение 3 «Описание функциональности классов серверной части» 20](#Приложение3)

[Приложение 4 «Описание полей, свойств и методов классов серверной части» 21](#Приложение4)

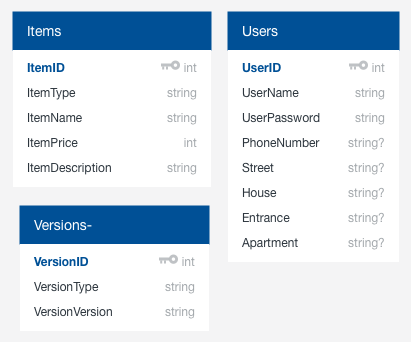
[Приложение 5 «Описание функциональности классов клиентской части» 27](#Приложение5)

[Приложение 6 «Описание полей, свойств и методов классов клиентской части» 2](#Приложение6)8

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 3](#ЛИСТРЕГИСТРАЦИИИЗМЕНЕНИЙ)3

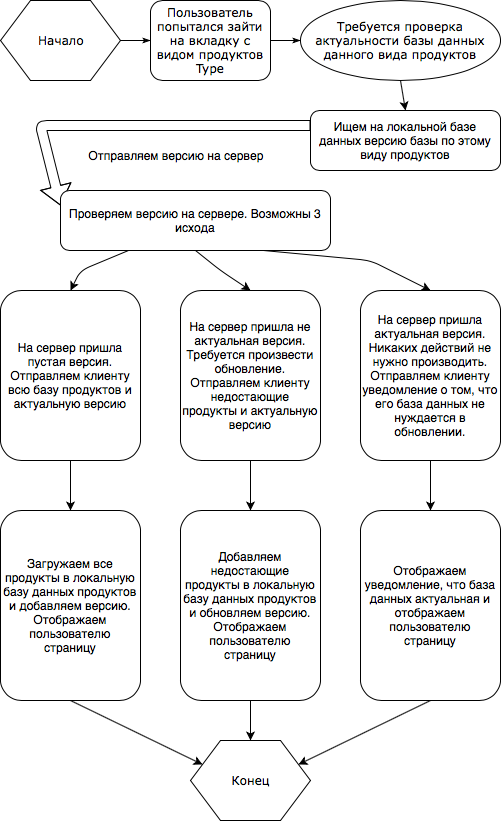
1. **ВВЕДЕНИЕ**
   1. **Наименование программы**  
      Наименование программы: «IOS-Приложение для заказа еды» («IOS-Application for Ordering Food»). Краткое название: «SushiApp».
   2. **Документы на основании которых ведется разработка**  
      Приказ декана факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики» № 2.3-02/1212-01 от 12.12.2017 «Об утверждении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук».
2. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**
   1. **Функциональное назначение**  
      Функциональное назначение программы - реализация мобильного IOS-приложения для предоставления пользователю удобной системы составления уникального заказа и оформления доставки по заданному адресу.
   2. **Эксплуатационное назначение**  
      Программа предназначена для любого пользователя, желающего составить заказ еды. Клиентская часть программы выполняется на мобильном устройстве пользователя.
3. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи на разработку программы**  
       Требуется реализовать мобильное IOS-приложение, которое представляет из себя удобною систему составления уникального заказа по меню, указанному в программном документе «Техническое задание», и оформления доставки по заданному адресу. Также должна быть возможность регистрации, авторизации в приложении, сохранении данных о доставке в системе.
   2. **Описание алгоритма и функционирования программы**  
       В приложении есть две основополагающие части: клиентская и серверная. Некоторые классы расположены, как и на клиентской стороне, так и на серверной, однако их функционал сведен к минимуму требуемому на каждой из частей приложения.

В разработке приложения использовалась база данных. Привожу ее структуру в виде изображения:



*Рис. 1 – Структура серверной базы данных*

* + 1. **Алгоритм загрузки продуктов с сервера**   
       Основной алгоритм, который делает эффективной загрузку товаров с сервера. Описание приведено в виде блок-схемы:



*Рис. 2 – Схема работы приложения согласно главному алгоритму*

* + 1. **Реализация серверной части приложения**  
       К серверной части приложения подключена сторонняя библиотека «NewtonsoftJson» ([9]), являющаяся передовым инструментом для разработки программ, требующих использовать сериализацию или десериализацию данных. Также используется пакет библиотек для реализации архитектуры сервера, таких как «Microsoft.AspNet.WebApi» ([10]) и «Microsoft.Owin» ([11]). Последний сторонний подключенный пакет - библиотека «System.Data.SqlClient» ([12]), предназначенная для базы данных. В программе реализован контроллер для каждой из операций приема данных с клиентской части программы: LoadDB\Login\Order\Registration\VersionChecker Controller. Остальные классы описывают объекты, которые используются для передачи данных мобильному приложению, а также заполнения таблиц базы данных.

*Ниже изложен список классов, использованных на серверной части программного продукта.*

– **Класс LoadDBController**

Контроллер, унаследованный от абстрактного класса ApiController. Реализует методы взаимодействия клиента с сервером и проводит операции, связанные с загрузкой лишь тех продуктов, которые потребовались пользователю во время работы с приложением. Именно этот класс обеспечивает эффективное распределение ресурсов для максимальной эффективности загрузки продуктов.

– **Класс LoginController**

Контроллер, унаследованный от абстрактного класса ApiController. Реализует методы взаимодействия клиента с сервером и проводит операции, связанные с авторизацией пользователя на сервере.

– **Класс OrderController**

Контроллер, унаследованный от абстрактного класса ApiController. Реализует методы взаимодействия клиента с сервером и проводит операции, связанные с оформлением доставки, сохранением дополнительной информации, а также выводом окончательно сформированного заказа на экран.

– **Класс RegistrationController**  
Контроллер, унаследованный от абстрактного класса ApiController. Реализует методы взаимодействия клиента с сервером и проводит операции, связанные с регистрацией пользователя на сервере.

– **Класс VersionCheckerController**  
Контроллер, унаследованный от абстрактного класса ApiController. Реализует методы взаимодействия клиента с сервером и проводит операции, связанные с проверкой актуальности локальной базы данных, расположенной на мобильном устройстве клиента.

– **Класс GSettings**  
Класс, в котором хранятся основные данные о подключении к серверу базы данных, а также о расположении сервера.

– **Класс PhotoConverter**

Позволяет перевести изображение, расположенное в директиве “SelfHostApiServer\PhotoDatabase\[type]\[name]”, где type – имя типа продукта, а name – название продукта, в код JSON для отправки клиенту на мобильное приложение. Все фотографии должны быть заранее размещены по указанному пути для верного функционирования приложения.  
– **Класс Program и Startup**Являются классами, в которых содержится метод Main(), отрабатывающий при запуске приложения. Требуются для первоначальной настройки сервера и его запуска.

* + 1. **Реализация клиентской части приложения**На клиентской части приложения, в свою очередь, также используется сторонняя библиотека «NewtonsoftJson» ([9]) для сериализации и десериализации передаваемых объектов. Также подключены библиотеки: «RestSharp» ([13]), использованная для отправки запросов на сервер, «SQLite-net-pcl» ([14]) для реализации архитектуры внутренней базы данных приложения. Интерфейс мобильного приложения разработан с помощью технологии Xamarin ([5] и [6]). Из-за этого дизайн приложения разработан на расширяемом языке разметки для приложений – языке «XAML», разработанным компанией Microsoft. Вся внутренняя логика приложения написана на языке программирования C# .NET.

*Ниже изложен список классов, использованных на клиентской части программного продукта*.

**Класс ItemDatabaseController**  
Используется для обращения к локальной базе данных продуктов мобильного приложения (выполняет функции добавления, удаления, обновления базы данных).

**Класс VersionDatabaseController**

Используется для обращения к локальной базе данных версий продуктов мобильного приложения (выполняет функции добавления, удаления, обновления базы данных).

**Класс RestService**

Класс, реализующий все используемые в программе методы для отправки данных на сервер. Производит все проверки отправляемых, а также получаемых данных.

**Класс Constants**   
Хранит данные о текущей сессии.

**Класс ShoppingCart**

Предоставляет доступ до корзины продуктов текущего сеанса.

*Ниже изложен список классов, использованных на как на клиентской, так и на серверной части*.

**Класс Item**

Используется для создания объекта продукта. Хранит идентификатор, название, цену, вид и изображение продукта.

**Класс Order**

Используется для создания объекта, хранящего всю информацию о заказе: логин пользователя, составлявшего заказ, его мобильный телефон, а также адрес доставки, корзину покупок и цену корзины.

**Класс User**

Используется для создания объекта, хранящего информацию о пользователе: его логин и пароль, и дополнительную сохраненную информацию, если таковая имеется.

**Класс DataVersion**   
Используется для создания объекта, хранящего информацию о виде продукта и его версии (актуальной в случае сервере или текущей на клиентской части программы).

* 1. **Описание и обоснование метода организации входных и выходных данных**  
      Элементы управления и поля ввода интерфейса клиентской части приложения позволяют пользователю формировать входные данные в ходе работы с программой.   
      Одна из самых удобных технологий для конвертирования данных перед передачей их на сервер – сериализация объектов в код JSON. Поэтому данные между клиентом и сервером в обе стороны передаются в формате текста JSON. Этот метод имеет преимущества в быстроте работы, удобности в реализации сериализации и десериализации объектов, компактности хранения данных над своими аналогами.  
      Соединение между системой Клиент-Сервер реализовано с помощью протокола HTTP (Hyper Text Transfer Protocol) ([4]). Этот протокол является наиболее популярным, ведь используется повсеместно во время работы с многими интернет ресурсами.
  2. **Описание и обоснование выбора технических и программных средств**  
      Для разработки интерфейса мобильного приложения выбрана технология Xamarin ([5] и [6]), позволяющая вести кроссплатформенную разработку программного обеспечения для мобильных устройств. В связке с этой технологией разработчиками подразумевается использование языка программирования **C# .NET** ([15]).   
      Также для реализации всей функциональности сервера требуется соответствующая технология **ASP.NET** ([10]), ее реализация также приведена на языке программирования **C# .NET** ([15]). Для этой технологии были подключены сторонние библиотеки к серверной части программы, такие как **Microsoft.Owin** ([11]) и **Microsoft.AspNet.WebApi** ([10]), а на клиентской части приложения подключены **Net.Http** ([4]) и **Xam.Plugin.Connectivity** ([6]) для взаимодействия с сервером.  
      В приложении также используются базы данных. На клиентской части использована **SQLite** ([14]), для реализации локальной базы данных, а на сервере используется библиотека **Data.SqlClient** ([12]).
  3. **Состав рекомендуемых технических и программных средств** Ниже предоставлен рекомендованный состав технических и программных средств для достижения быстродействия в работе программного обеспечения:

**Клиентская часть:**– Устройство, требуемое для эксплуатации, - мобильный телефон iPhone 6 или новее с установленной IOS 8.3 или новее;

– Также можно использовать симулятор мобильного телефона iPhone с персонального компьютера. В таком случае рекомендуется такое устройство:

– Операционная система High Sierra, версия 10.13.4;

– Процессор с тактовой частотой 3.19 ГГц;

– Объем ОЗУ 16 ГБ;

– Свободное количество памяти – 500 Мб;

– Стабильное соединение с интернетом с скоростью 10 Мбит\сек;

**Серверная часть:**  
– Для поддержания работы у одной тысячи пользователей рекомендуется иметь 1 Гб ОЗУ. При большем числе пользователей рекомендовано увеличивать количество ОЗУ на 1 Гб на каждую тысячу пользователей.;

– Рекомендовано иметь больше, чем 200 Мб свободного места на жестком диске или другом носителе устройства;

– Рекомендуется иметь стабильное Интернет-соединение с скоростью в 150 Мбит\сек на каждых 250 пользователей;  
– Sql Operations Studio для настройки базы данных для ОС OSX или Microsoft SQL Server 2015 или новее для ОС Windows;

– Программа Docker для ОС OSX;

– Серверная работа должна быть настроена на постоянную работу (без ухода в спящий режим, например) и должна быть постоянно активна.

1. **ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**
   1. **Ориентировочная экономическая эффективность** Данная работа не предусматривает оценки экономической эффективности разработанного программного обеспечения.
   2. **Предполагаемая потребность** Разработанный продукт обеспечит удобство в оформлении заказа пользователями у компании «Отличные Суши. Москва» с помощью мобильного приложения.
   3. **Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами** На момент начала разработки приложения на рынке не было выявлено продукта, поддерживающего меню продуктов, указанное в программном документе «Техническое задание». Приложений на рынке, имеющих схожий функционал на рынке много.  
       Самое известное приложение на мобильный телефон, поддерживающий операционную систему IOS, является приложение «Delivery Club», разработанное компанией «Mail.ru Group» в 2009 году. Оно поддерживает большое количество загружаемых продуктов, однако не отличается быстродействием из-за своего масштаба.  
       При разработке приложения был сделан упор на максимальную оптимизацию скорости загрузки товаров с сервера, а также реализации всех требуемых функций для реализации удобной системы составления заказа по указанному меню.   
       Разработанный продукт обеспечивает более быстродействующую систему, чем существующие аналоги, в основном, за счет меньшего количества обрабатываемых приложением продуктов.
2. **ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ**
3. ГОСТ 19.404-79. Пояснительная записка. // Единая система программной документации. – Москва: Издательство стандартов, 2005.
4. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. Полное меню продуктов. // Отличные суши, Москва (http://www.great-sushi.ru/).
6. Http протокол. // Википедия (https://ru.wikipedia.org/wiki/HTTP).
7. Основы Xamarin.Forms [Electronic resource] // Metanit.com [Official website]. URL: https://metanit.com/sharp/xamarin/ (accessed: 10.05.2018).
8. Документация по Xamarin [Electronic resource] // Microsoft Docs [Official website]. URL: https://docs.microsoft.com/ru-ru/xamarin/ (accessed: 19.05.2018).
9. ГОСТ 19.106-78. Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – Москва: Издательство стандартов, 2005.
10. ГОСТ 19.301-78. Программа и методика испытаний. // Единая система программной документации. – Москва: Издательство стандартов, 2005.
11. Документация по NewtonJson [Electronic resource] // Json.Net Documentation [Official website]. URL: https://www.newtonsoft.com/json/help/html/Introduction.html (accessed: 20.05.2018).
12. Документация по Asp.Net Web Api [Electronic resource] // Microsoft Docs [Official website]. URL: https://docs.microsoft.com/en-us/aspnet/web-api/ (accessed: 20.05.2018).
13. Документация по .NET интерфейсу OWIN [Electronic resource] // Microsoft Docs [Official website]. URL: https://docs.microsoft.com/ru-ru/aspnet/core/fundamentals/owin/ (accessed: 20.05.2018).
14. Документация по System.Data.Client [Electronic resource] // MDSN [Official website]. URL: https://mdsn.microsoft.com/ru-ru/library/system.data.sqlclient (accessed: 20.05.2018).
15. Документация по RestSharp [Electronic resource] // GitHub [Official website]. URL: https://github.com/restsharp/RestSharp/wiki (accessed: 20.05.2018).
16. Документация по Sql-lite-pcl [Electronic resource] // GitHub [Official website]. URL: https://github.com/praeclarum/sqlite-net (accessed: 20.05.2018).
17. Документация по C# .NET [Electronic resource] // Microsoft Docs [Official website]. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (accessed: 20.05.2018).

**Приложение 1**

**Описание функциональности классов, использованных на серверной и на клиентской части приложения**

|  |  |
| --- | --- |
| **Имя класса** | **Назначение** |
| Item | Предназначен для хранения информации о продукте. |
| Order | Предназначен для хранения информации о заказе. |
| DataVersion | Предназначен для хранения информации о хранимой версии продуктов некоего вида. |
| User | Предназначен для хранения информации о пользователе. |

**Приложение 2**

**Описание полей, свойств и методов, использованных в классах на серверной и клиентской части**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс Item** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| Id | public | int | Идентификатор продукта. Также для хранения в базе данных имеет ключи [AutoIncrement], [PrimaryKey], [NotNull]. | |
| Name | public | string | Название продукта | |
| Type | public | string | Тип продукта | |
| Price | public | int | Цена продукта | |
| Description | public | string | Описание продукта | |
| Pic | public | byte[] | Массив битов, хранящий изображение продукта | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| Equals | public | bool | Item other | Реализует интерфейс IEquatable<T> и сравнивает продукты по характеристикам. |
| ToString | public | string | - | Перегрузка стандартного метода объекта в C#. Возвращает строку с характеристиками продукта. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс Order** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| UserName | public | string | Имя пользователя, сделавшего заказ. | |
| UserPassword | public | string | Пароль пользователя, сделавшего заказ | |
| Phone | public | string | Телефон, указанный в заказе | |
| Street | public | int | Улица, указанная в заказе | |
| House | public | string | Дом, указанный в заказе | |
| Entrance | public | string | Подъезд, указанный в заказе | |
| Apartment | public | string | Квартира, указанная в заказе | |
| Saver | public | bool | Ключ, характеризующий необходимость сохранения у пользователя данных о заказе | |
| OrderList | public | List<Item> | Коллекция продуктов, которые заказывает пользователь. | |
| Price | public | int | Общая цена заказа | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| ToString | public | string | - | Перегрузка стандартного метода объекта в C#. Возвращает строку с данными о заказе. |
| GetItemsFromListView | private | List<Item> | ObservableCollection  <ShoppingItem> | Возвращает список продуктов, конвертируя переданный в метод список предметов корзины, использующийся для отображения на вкладке «Корзина покупок». |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс DataVersion** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| Id | public | int | Идентификатор версии, также на клиентской части присутствуют ключи [AutoIncrement], [PrimaryKey], [NotNull]. | |
| TypeName | public | string | Название вида продуктов, версия которых хранится в локальной базе данных | |
| Version | public | string | Версия вида продуктов | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| ToString | public | string | - | Перегрузка стандартного метода объекта в C#. Возвращает строку с данными о версии. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс User** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| Id | public | int | Идентификатор пользователя. | |
| UserName | public | string | Логин пользователя. | |
| Password | public | string | Пароль пользователя. | |
| PhoneNum | public | string | Указанный мобильный телефон пользователя. | |
| Street | public | string | Указанная улица пользователя | |
| House | public | int | Указанный дом пользователя | |
| Entrance | public | string | Указанный подъезд пользователя | |
| Apartment | public | string | Указанная квартира пользователя | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| CheckAuthorizationInfo | public | bool | - | Проверяет авторизирован ли пользователь в приложении. |
| ToString | public | string | - | Перегрузка стандартного метода объекта в C#. Возвращает строку с данными о пользователе. |
| Equals | public | bool | User other | Реализует интерфейс IEquatable<T>. Сравнивает объекты пользователей по логинам и паролям. |

**Приложение 3**

**Описание функциональности классов, использованных на серверной части приложения**

|  |  |
| --- | --- |
| **Имя класса** | **Назначение** |
| LoadDBController | Контроллер для загрузки продуктов с базы данных и приеме\отправке запросов. |
| LoginController | Контроллер для авторизации пользователя на сервере и приеме\отправке запросов. |
| OrderController | Контроллер для оформления заказа и приеме\отправке запросов. |
| RegistrationController | Контроллер для регистрации пользователя и приеме\отправке запросов. |
| VersionCheckerController | Контроллер для проверки актуальности версии и приеме\отправке запросов. |
| GSettings | Хранит общие сведения о подключении к базе данных. |
| Program | Основной класс, запускающий сервер. |
| Startup | Настраивает параметры, характеризующие запущенный сервер. |

**Приложение 4**

**Описание полей, свойств и методов, использованных в классах на серверной части**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс LoadDBController** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| photoConverter | private | PhotoConverter | Объект класса, позволяющего конвертировать изображения в массив битов для передачи клиенту. | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| [HttpGet] Get | public | IHttpActionResult | string productType | Получает тип продуктов в качестве параметров и возвращает список продуктов, которые нужно добавить клиенту на локальную базу данных. |
| GetItems | private | List<Item> | string productType | Возвращает список продуктов, загруженных из актуальной базы данных сервера. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс LoginController** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| [HttpGet] Get | public | IHttpActionResult | string userInfo | Получает пользователя и пытается проверить наличие такого пользователя в системе. Возвращает значения юзера, если таков был найден. Иначе возвращает null. |
| CheckForUser | private | User | User loggined | Проверяет на наличие в базе данных переданного в качестве параметра пользователя и возвращает найденного. |
| CheckAddInfo | private | bool | SqlDataReader reader | Проверяет, имеет ли пользователь сохраненные данные о заказе в базе данных. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс OrderController** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| [HttpGet] Get | public | IHttpActionResult | string orderItems | Получает на вход заказ и выводит информацию на сервер, если заказ был проверен. Отправляет клиенту уведомление об успешности проведенной операции. |
| SaveUserInfo | private | bool | Order order | Сохраняет в базе данных данные о заказе, если соответствующее условие было поставлено. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс RegistrationController** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| [HttpGet] Get | public | IHttpActionResult | string registrationUser | Получает на вход информацию о пользователе и пытается проверить, нет ли уже такого пользователя в базе данных. |
| CheckForUser | private | bool | User registrated | Проверяет на наличие в базе данных переданного в качестве параметра пользователя и возвращает найден ли такой пользователь или нет. |
| RegisterUser  InDatabase | private | int | User user | Рассчитывает и возвращает новый идентификатор для пользователя. |
| GetUsersCount | private | int | - | Возвращает общее количество зарегистрированных пользователей. |
| RegisterUser | private | void | Int id, User user | Добавляет нового пользователя в базу данных сервера. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс VersionCheckerController** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| [HttpGet] Get | public | IHttpActionResult | string data | Получает версию данных клиента и проверяет ее на актуальность, возвращает новую версию, если версия клиента не актуальна. |
| GetVersion | private | string | string type | Обращается к базе данных и возвращает актуальную версию данного вида продуктов. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс GSettings** | | | |
| **Поля и свойства** | | | |
| Имя | Модификатор доступа | Тип | Описание |
| photoConverter | private | PhotoConverter | Объект класса, позволяющего конвертировать изображения в массив битов для передачи клиенту. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс Program** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| ServerStart | private | void | - | Запускает сервер по заданному адресу |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс Startup** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| Configuration | public | void | IAppBuilder app | Создает все настройки для запуска сервера. |

**Приложение 5**

**Описание функциональности классов, использованных на клиентской части приложения**

|  |  |
| --- | --- |
| **Имя класса** | **Назначение** |
| ItemDatabaseController | Контроллер для работы с локальной базой данных продуктов. |
| VersionDatabaseController | Контроллер для работы с локальной базой данных версий продуктов. |
| RestService | Класс для отправки запросов на сервер. |
| Constants | Класс для хранения статичных данных в ходе работы с приложением. |
| ShoppingCart | Класс для хранения данных о корзине покупок. |

**Приложение 6**

**Описание полей, свойств и методов, использованных в классах на клиентской части**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс ItemDatabaseController** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| database | private | SQLiteConnection | Объект подключения к локальной базе данных. | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| GetItem | public | Item | Int id | Возвращает продукт по указанному идентификатору. |
| SaveListOfItems | public | int | List<Item> list | Сохраняет все переданные продукты в базе данных и возвращает количество добавленных продуктов. |
| SaveItem | public | int | Item item | Сохраняет переданный продукт в базе данных. |
| DeleteItem | public | int | Int id | Удаляет продукт из базы данных по указанному идентификатору. |
| GetCountOfItems | public | int | - | Возвращает количество продуктов в базе данных. |
| DeleteAllTypeItems | public | int | String type | Удаляет все продукты с указанным типом. |
| DeleteWholeDB | public | int | - | Удаляет всю базу данных продуктов. |
| GetItemsOfType | public | List<Item> | String type | Возвращает список продуктов указанного типа |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс VersionDatabaseController** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| database | private | SQLiteConnection | Объект подключения к локальной базе данных. | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| GetVersionByID | public | Item | Int id | Возвращает версию по указанному идентификатору. |
| SaveVersion | public | int | DataVersion data | Сохраняет переданную версию в базе данных. |
| GetTypeVersion | public | string | String type | Возвращает версию из базы данных по указанному типу продуктов. |
| ChangeDataVersion | public | bool | String type, string version | Изменяет в базе данных версию продуктов данного типа. |
| SaveDataVersion | public | void | String type, string version | Добавляет версию в базу данных. |
| DeleteWholeDB | public | int | - | Удаляет всю базу данных версий. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс RestService** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| client | private | RestClient | Объект для отправки запросов с нужными параметрами. | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| CheckConnection | private | bool | - | Возвращает значение булевского типа, есть ли подключение к интернету в данный момент. |
| Login | public | Task<string> | User loginned | Отправляет запрос авторизации на сервере на контроллер LoginController. |
| Register | public | Task<string> | User registered | Отправляет запрос регистрации на сервере на контроллер RegistrationController. |
| CheckDataTypeVersion | public | Task<string> | String type, string version | Отправляет запрос проверки актуальности версии базы данных на сервере на контроллере VersionCheckerController. |
| LoadDBofPageProducts | public | Task<int> | String type | Отправляет запрос загрузки продуктов с сервера на контроллер LoadDBController. |
| UpdateDBofPageProducts | public | Task<int> | String type | Отправляет запрос обновления версии и загрузки новых продуктов с сервера на контроллер LoadDBController. |
| SendOrder | public | Task<string> | Order order | Отправляет запрос оформления заказа на сервер на контроллер OrderController. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс Constants** | | | |
| **Поля и свойства** | | | |
| Имя | Модификатор доступа | Тип | Описание |
| MainUser | public | User | Хранит авторизированного пользователя. |
| serverSettings | public | ServerSettings | Хранит данные об адресе сервера. |
| BackgroundColor | public | Xamarin.Forms.Color | Хранит стандартный цвет заднего фона. |
| MainTextColor | public | Xamarin.Forms.Color | Хранит стандартный цвет текста. |
| LoginIconHeight | public | int | Хранит стандартную высоту иконки. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Класс Constants** | | | | | | | |
| **Поля и свойства** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | Описание | |
| Cart | public | | ObservableCollection<  ShoppingListViewItem> | | | Коллекция продуктов корзины. | |
| Price | public | | int | | | Цена корзины. | |
| **Методы** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | Параметры | | Описание |
| GetPriceValue | | Private | | int | - | | Производит расчет общей цены корзины. |
| ClearAll | | Public | | void | - | | Очищает корзину. |

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |