Министерство образования Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**КУРСОВОЙ ПРОЕКТ**

по дисциплине **«Модульное тестирование»**

на тему **«Электронный магазин для компьютерных комплектующих»**

Выполнил:

студент группы 10701116 Багиров Р.А.

Руководитель: Попова Ю.Б.

Минск 2018

Министерство образования Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

по дисциплине **«Модульное тестирование»**

на тему **«Электронный магазин для компьютерных комплектующих»**

Выполнил:

студент группы 10701116 Багиров Р.А.

Руководитель: Попова Ю.Б.

Минск 2018

Введение 4

1. Требования к программному продукту 7

1.1. Регистрация 7

1.2. Аутентификация 9

1.3. Администратор 9

1.4. Авторизованный пользователь 11

1.5. Неавторизованный пользователь 13

2. Проектирование программных модулей 15

3. Реализация программных модулей 16

3.1. Архитектура ПО и описание протокола взаимодействия клиента и сервера 16

3.2. Серверная часть 17

3.3. Клиентская часть 19

4. Проектирование модульных тестов 21

4.1. Построение ориентированных графов Мак-кейба 21

4.2. Формирование проходов 22

5. Реализация модульного тестирования 26

5.1. Реализация автоматизированных скриптов 26

5.2. Анализ результатов тестов 26

5.3. Отладка тестов 26

Заключение 27

Список использованных источников 28

ПРИЛОЖЕНИЕ А 29

Листинг исходных кодов 29

ПРИЛОЖЕНИЕ Б 30

Листинг автоматизированных скриптов 30

# Введение

Компьютерной (вычислительной) сетью называется совокупность компьютеров (ЭВМ), взаимосвязанных через каналы передачи данных и обеспечивающих пользователей средствами обмена информацией и коллективного использования аппаратных, программных и информационных ресурсов сети.

История развития компьютерных сетей началась в 60-х годах прошлого столетия. Сначала появились многотерминальные системы разделения времени. В таких системах мощная ЭВМ обслуживала одновременно несколько пользователей, имеющих в своем распоряжении терминал (монитор с клавиатурой), с помощью которого он мог вести диалог с ЭВМ. ЭВМ по очереди обрабатывала программы и данные, поступающие с каждого терминала. Терминалы, как правило, рассредоточивались по всему предприятию и функции ввода-вывода информации были распределенными, а ее обработка проводилась только центральной ЭВМ. Подобные многотерминальные централизованные системы внешне напоминали локальные вычислительные сети, до создания которых в действительности нужно было пройти еще большой путь. Затем была решена задача доступа к ЭВМ с терминалов, удаленных от нее на сотни (а то и тысячи) километров. Терминалы в этом случае соединялись с ЭВМ через телефонные линии с помощью модемов. Такие вычислительные сети получили название распределенных или глобальных. Следующим этапом в развитии вычислительных сетей стали соединения не только «терминал — ЭВМ», но и «ЭВМ — ЭВМ». ЭВМ стали обмениваться данными в автоматическом режиме и впервые появились возможности обмена файлами, синхронизации баз данных, использования электронной почты. Иными словами, появились те службы, которые в настоящее время стали традиционными сетевыми сервисами.

Исторически первые компьютерные сети были созданы агентством АRРА по заданию военного ведомства США. В 1969 году министерство обороны США инициировало работы по объединению в единую сеть суперкомпьютеров оборонных и научно-исследовательских центров. Эта сеть, получившая название ARPANET, стала отправной точкой для создания самой известной ныне глобальной сети — Internet.

В 70-х гг. XX века, в связи с развитием микроэлектроники, начался интенсивный процесс распределения вычислительных ресурсов, что впоследствии привело к необходимости обратного объединения всех вычислительных ресурсов в одну систему. Только теперь это объединение происходило уже не на базе одного компьютера, а путем подключения к сети отдельных распределенных компьютеров. Подобные компьютерные сети стали называться локальными компьютерными сетями.

На начальном этапе создания локальных компьютерных сетей для объединения компьютеров использовались самые разнообразные нестандартизированные устройства и программное обеспечение. Создание сети в это время требовало от разработчиков изобретательности и больших усилий. В середине 80-х гг. положение дел в локальных компьютерных сетях стало кардинально меняться в сторону создания стандартных технологий объединения компьютеров в единую сеть. Были разработаны специальные методы и правила обмена информацией между компьютерами, среди которых наиболее известными стали стандарты Ethernet, Toking Ring, FDDI, Arcnet. В этих стандартах были строго регламентированы длина, вид и порядок следования кодов, посылаемых компьютерами в сеть, правила доступа к сети отдельных компьютеров и т.д. Кроме того, в это время начали интенсивно использоваться стандартные персональные компьютеры, которые очень быстро потеснили мини-ЭВМ и мэйнфреймы. Разработанные стандартные сетевые технологии, а также использование персональных компьютеров значительно упростили процесс создания компьютерных сетей. Для создания сети достаточно стало приобрести специальные сетевые платы (сетевые адаптеры) соответствующего стандарта, стандартный кабель со стандартными разъемами и установить на компьютер сетевую операционную систему.

На сегодняшний день основными направлениями использования компьютерных сетей являются следующие:

1. Совместный доступ к аппаратным, программным и информационным ресурсам (использование дисков или только определенных папок и файлов других компьютеров, принтеров, программного обеспечения, баз и банков данных);
2. Предоставление коммуникационных услуг (службы информации, электронная почта, телеконференции и т.д.);
3. Распределенная обработка данных (сети можно использовать для обработки данных на отдельных компьютерах, связанных между собой и представляющих распределенную систему).

В общем случае любая компьютерная сеть состоит из набора трех основных компонентов:

* рабочих станций (персональных компьютеров пользователей);
* файлового сервера (главного компьютера сети);
* сети или средств передачи данных (физической передающей среды и АПД), обеспечивающих обмен информацией между компьютерами.

В частном случае компьютерная сеть может содержать несколько серверов разной степени иерархии или состоять только из двух или более однотипных рабочих станций.

Функциональные возможности сети определяются услугами, которые она предоставляет. Для реализации каждой из услуг сети и доступа пользователя к этой услуге используется сетевое программное обеспечение.

В настоящее время распространены две основные концепции построения сетевого ПО.

Первая концепция ориентирована на предоставление многим пользователям ресурсов главного компьютера сети — файлового сервера. Управление ресурсами файлового сервера и предоставление к ним доступа производится сетевой операционной системой. Ее основная часть находится на файловом сервере, а на рабочих станциях (компьютерах пользователей) устанавливается только небольшая оболочка, выполняющая роль интерфейса между программами, обращающимися за ресурсами к файловому серверу. Рабочие станции используют программы или данные файлового сервера, а также другие его ресурсы (принтер, модем и т.п.). Программы файлового сервера могут использоваться всеми пользователями одновременно, но для выполнения модули этих программ по мере необходимости переносятся на рабочую станцию. При этом вся обработка данных, даже если они являются общими ресурсами и хранятся на файловом сервере, происходит непосредственно на рабочих станциях (очевидно, что для этого файлы, в которых хранятся данные, должны быть перемещены на рабочую станцию).

Во второй концепции, называемой архитектурой «клиент-сервер», ПО не только обеспечивает коллективное использование ресурсов, но и ориентировано на их обработку в местах размещения ресурсов по запросам пользователей. Программные системы архитектуры клиент-сервер состоят из двух частей: программного обеспечения сервера и программного обеспечения пользователя-клиента. Работа организуется следующим образом: программы-клиенты выполняются на компьютере пользователя и посылают запросы к программе-серверу, которая работает на компьютере общего доступа. Основная обработка данных производится мощным сервером, а на компьютер пользователя посылаются только результаты выполнения запроса.

В приложениях глобальных сетей архитектура клиент-сервер является основной. Широко известны Web-серверы, обеспечивающие хранение и обработку гипертекстовых страниц, FTP-серверы, серверы электронной почты и множество других. Клиентские программы перечисленных служб позволяют сформулировать запрос на получение услуги со стороны этих серверов и принять от них ответ.

# Требования к программному продукту

Система предназначена для заказа компьютерных комплектующих через Интернет.

В системе предусмотрено три роли пользователей:

* Администратор – зарегистрированный пользователь, имеющий доступ к серверной части приложения
* Пользователь – зарегистрированный пользователь, имеющий доступ только к клиентской части приложения
* Гость – незарегистрированный пользователь, имеющий доступ только к клиентской части приложения

Функции администратора: выйти, добавление категорий, удаление категорий, добавление товаров, удаление товаров, редактирование товаров, поиск пользователя по логину, просмотр списка пользователей (логин и история покупок), удаление пользователей, просмотр списка заказов, подтверждение заказа, просмотр всей истории заказов, удаление записи из истории заказов, очистка истории заказов.

Функции пользователя: выйти, просмотр истории покупок, удаление записи из истории, очистка истории, сортировка товаров, поиск товаров по критериям, просмотр подробно о товаре, добавление товара в корзину, просмотр корзины, удаление товара из корзины, оформление заказа (адрес, мобильный телефон).

Функции гостя: регистрация (логин и пароль), авторизация, сортировка товаров, поиск товаров по критериям, просмотр подробно о товаре, добавление товара в корзину, просмотр корзины, удаление товара из корзины, оформление заказа (адрес, мобильный телефон).

## Регистрация

|  |  |
| --- | --- |
| R1.1 | Для создания нового аккаунта используется страница регистрации (registration.php)    На странице содержатся следующие элементы:   * Поле для ввода логина * Поле для ввода почты * Поле для ввода пароля * Поле для подтверждения пароля * Кнопка для регистрации на сайте |
| R1.2 | Все поля («Логин», «Почта», «Пароль» и «Подтверждение пароля») являются обязательными для ввода. |
| R1.3 | Логин может содержать только буквы латинского алфавита в нижнем регистре, цифры, символы «\_» и «-». Допустимый размер логина – от 5 до 64 символов (включительно). |
| R1.4 | Если пользователь ввел логин, который не отвечает требованиям «R1.3», то после нажатия на кнопку «Зарегистрироваться» появится сообщение «Некорректно введен логин». |
| R1.5 | Если пользователь ввел логин, который уже занят, то при нажатии на кнопку «Зарегистрироваться» появится сообщение «Данный логин уже занят». |
| R1.6 | Почта должна соответствовать формату «a@b.c», где части «a» и «b» могут содержать только буквы латинского алфавита в нижнем регистре, цифры, символы «\_», «.» и «-». Часть «с» может содержать только буквы латинского алфавита в нижнем регистре и символ «.». Часть «с» может иметь размер не менее двух символов и не более шести.  Длина почты не должна превышать 128 символов. |
| R1.7 | Если пользователь ввел почту, которая не отвечает требованиям «R1.6», то после нажатия на кнопку «Зарегистрироваться» появится сообщение «Некорректно введен email». |
| R1.8 | Если пользователь ввел почту, которая привязана к другому зарегистрированному пользователю, то после нажатия на кнопку «Зарегистрироваться» появится сообщение «Пользователь с данной почтой уже существует». |
| R1.9 | Пароль может содержать любые символы. Допустимый размер пароля – от 5 до 32 символов. |
| R1.10 | Если пользователь ввел пароль, который не отвечает требованиям «R1.9», то после нажатия на кнопку «Зарегистрироваться» появится сообщение «Некорректно введен пароль». |
| R1.11 | Если пароли в полях «Пароль» и «Подтверждение пароля» не совпадают, то после нажатия на кнопку «Зарегистрироваться» появится сообщение «Неверно подтвержден пароль». |
| R1.12 | Если пользователь заполнил все поля корректно, то после нажатия на кнопку «Зарегистрироваться» появится новый аккаунт в системе и пользователя перенаправит на главную страницу (index.php). |

## Аутентификация

|  |  |
| --- | --- |
| R2.1 | Для аутентификации используется страница authentication.php    На странице содержатся следующие элементы:   * Поле для ввода логина * Поле для ввода пароля * Кнопка для входа |
| R2.2 | Все поля («Логин» и «Пароль») являются обязательными для ввода |
| R2.3 | Логин может содержать только буквы латинского алфавита в нижнем регистре, цифры, символы «\_» и «-». Допустимый размер логина – от 5 до 64 символов (включительно). |
| R2.4 | Если пользователь ввел логин, который не отвечает требованиям «R2.3», то при нажатии на кнопку «Войти» появится сообщение «Некорректно введен логин». |
| R2.5 | Пароль может содержать любые символы. Допустимый размер пароля – от 5 до 32 символов. |
| R2.6 | Если пользователь ввел пароль, который не отвечает требованиям «R2.5», то при нажатии на кнопку «Войти» появится сообщение «Некорректно введен пароль». |
| R2.7 | Если пользователь ввел несуществующий логин или не подходящий пароль, то при нажатии на кнопку «Войти» появится сообщение «Неправильный логин или пароль». |
| R2.8 | Если пользователь заполнил все поля корректно, то после нажатия на кнопку «Войти» пользователя перенаправит на главную страницу (index.php) или на страницу администратора (admin.php), если данный пользователь является администратором (имеет логин «admin»). |

## Администратор

**Общие сведения**

Данную часть системы может видеть только авторизованный администратор. Администратор может просматривать список пользователей, историю заказов выбранного пользователя. Администратор может удалить пользователя. Администратор может добавить или удалить категории. Администратор может добавлять, удалять и редактировать товары. Также администратор может просмотреть список заказов, подтвердить заказ, удалить заказ, просмотреть всю историю заказов, удалить запись из истории заказов, очистить историю заказов. Администратор может выйти.

|  |  |
| --- | --- |
| R3.1 | На главной странице администратора (admin.php) расположено понятное пользовательское меню в виде ссылок: |
| R3.2 | После нажатия на ссылку «Выйти» администратора перенаправляет на главную страницу (index.php). |
| R3.3 | После нажатия на ссылку «Список пользователей» открывается страница users.php |
| R3.4 | На странице users.php имеется поле для поиска по логину и кнопка «Поиск», а также таблица со всеми пользователями. |
| R3.5 | Поиск выдает всех пользователей, чей логин содержит подстроку, которая соответствует строке в поле для поиска. |
| R3.6 | Если поиск ничего не вернул, то на месте таблицы будет надпись «Пользователей с таким логином не найдено». |
| R3.7 | Таблица с пользователями содержит логин и две кнопки – «История заказов» и «Удалить». |
| R3.8 | После нажатия на кнопку «История заказов» напротив логина выбранного пользователя открывается страница userHistory.php с историей заказов выбранного пользователя. |
| R3.9 | На странице userHistory.php имеется таблица с историей заказов выбранного пользователя. |
| R3.10 | Напротив записи на странице userHistory.php имеется кнопка «Удалить», которая удалит запись, если на нее нажать. |
| R3.11 | После нажатия на кнопку «Удалить» напротив логина выбранного пользователя выбранный пользователь будет удален. |
| R3.12 | После нажатия на ссылку «Список заказов» открывается страница «order.php» с таблицей заказов. |
| R3.13 | Напротив записи на странице order.php имеется кнопка «Подтвердить» и «Удалить». |
| R3.14 | После нажатия на кнопку «Подтвердить» напротив заказа заказ подтверждается. |
| R3.15 | После нажатия на кнопку «Удалить» напротив заказа заказ удаляется. |
| R3.16 | После нажатия на ссылку «Список товаров» открывается страница adminGoods.php с таблицей товаров. |
| R3.17 | На странице adminGoods.php имеется поле для поиска по названию и кнопка «Поиск», а также таблица со всеми товарами. |
| R3.18 | Поиск выдает все товары, чье название содержит подстроку, которая соответствует строке в поле для поиска. |
| R3.19 | Если поиск ничего не вернул, то на месте таблицы будет надпись «Товаров с таким названием не найдено». |
| R3.20 | Таблица с товарами содержит название товара и две кнопки – «Изменить» и «Удалить». |
| R3.21 | После нажатия на кнопку «Изменить» открывается страница goodSet.php, которая содержит:   * Название товара * Цена товара * Описание * Изображение * Кнопка «Подтвердить» |
| R3.22 | Название товара не должно быть меньше пяти символов и не должно быть больше 128 символов. |
| R3.23 | Если администратор нажал на «подтвердить» и название товара не соответствует требованиям «R3.20», то появится сообщение «Некорректное название товара». |
| R3.24 | Цена товара должна быть целым неотрицательным числом, меньшим 2 147 483 648. |
| R3.25 | Если администратор нажал на «подтвердить» и цена товара не соответствует требованиям «R3.22», то появится сообщение «Некорректная цена товара». |
| R3.26 | Описание не является обязательным. Описание должно содержать не более 10000 символов. |
| R3.27 | Если администратор нажал на «подтвердить» и описание содержит более 10000 символов, то появится сообщение «Слишком большое описание». |
| R3.28 | Изображение является необязательным атрибутом. |
| R3.29 | Администратор может загружать изображения формата «.jpeg», «.jpg» и «.png». |
| R3.30 | Если все поля заполнены корректно, то после нажатия на кнопку «Подтвердить» принимаются изменения. |
| R3.31 | После нажатия на ссылку «Добавить товар» открывается страница addGoods.php со списком категорий. |
| R3.32 | После нажатия на одну из категорий открывается страница goodSet.php |
| R3.33 | Требования для страницы goodSet.php соответствуют требованиям «R3.19 – R3.28» |
| R3.34 | После нажатия на ссылку «Добавить категорию» открывается страница addCategories.php, которая содержит:   * Название категории * Изображение * Кнопка «Подтвердить» |
| R3.35 | Размер названия категории не должен быть меньше двух символов и не больше 64 символов. |
| R3.36 | Если администратор нажал на «подтвердить» и название категории не соответствует требованиям «R3.33», то появится сообщение «Некорректное название категории». |
| R3.37 | Категория (т.е. ее название) не должна дублироваться. |
| R3.38 | Если администратор нажал на «подтвердить» и название категории не соответствует требованиям «R3.37», то появится сообщение «Категория с таким названием уже существует». |
| R3.39 | Изображение является необязательным атрибутом. |
| R3.40 | Администратор может загружать изображения формата «.jpeg», «.jpg» и «.png». |
| R3.41 | Если все поля заполнены корректно, то после нажатия на кнопку «Подтвердить» создается новая категория. |
| R3.42 | После нажатия на ссылку «Удалить» открывается страница categories.php с таблицей категорий. |
| R3.43 | Напротив названия категории в таблице на странице categories.php имеется кнопка «Удалить». |
| R3.44 | После нажатия на кнопку «Удалить» выбранная категория будет удалена. Также будут «каскадно» удалены все товары, принадлежащие данной категории. |
| R3.45 | После нажатия на ссылку «История заказов» открывается страница history.php с таблицей историй заказов и кнопкой «очистить историю». |
| R3.46 | Напротив истории на странице history.php имеется кнопка «Удалить» |
| R3.47 | После нажатия на кнопку «Удалить» запись удаляется. |
| R3.48 | После нажатия на кнопку «Очистить историю» история будет полностью очищена. |

## Авторизованный пользователь

**Общие сведения**

Данную часть системы может видеть авторизованный пользователь. Пользователь может выбрать одну из категорий товара, просмотреть товары, которые имеются в базе данных, после чего он может добавить (удалить из) их в корзину и оформить заказ. Также пользователь может просматривать свою историю покупок и удалять из нее записи (или очистить всё).

|  |  |
| --- | --- |
| R4.1 | На главной странице index.php находится шапка и ссылки по категориям: |
| R4.2 | После нажатия на одну из категорий открывается страница goods.php c таблицей товаров, строкой поиска и кнопкой «Поиск», ссылками сортировки «по названию» и «по цене» и шапкой как в index.php. |
| R4.3 | Поиск выдает все товары, чьё название содержит подстроку, которая соответствует строке в поле для поиска. |
| R4.4 | Если поиск ничего не вернул, то на месте таблицы будет надпись «Товаров с таким названием не найдено». |
| R4.5 | Если нажать на ссылку сортировки «по названию», то все строки в таблице сортируются по названию. |
| R4.6 | Если еще раз нажать на ссылку сортировки «по названию», то все строки в таблице сортируются по названию в обратном порядке. |
| R4.7 | Если нажать на ссылку сортировки «по цене», то все строки в таблице сортируются по цене. |
| R4.8 | Если еще раз нажать на ссылку сортировки «по цене», то все строки в таблице сортируются по цене в обратном порядке. |
| R4.9 | После нажатия на название товара в таблице открывается страница good\_page.php c названием, ценой, изображением, описанием товара и кнопкой «Добавить в корзину» |
| R4.10 | После нажатия на кнопку «Добавить в корзину» товар добавляется в корзину. |
| R4.11 | “Пекарь” в шапке – это ссылка на главную страницу index.php |
| R4.12 | После нажатия на ссылку «История» открывается страница «userHistory.php» |
| R4.13 | На странице userHistory.php имеется таблица с историей заказов, кнопка «Очистить историю». |
| R4.14 | Напротив записи на странице userHistory.php имеется кнопка «Удалить», которая удалит запись, если на нее нажать. |
| R4.15 | После нажатия на кнопку «Очистить историю» история будет полностью очищена. |
| R4.16 | После нажатия на ссылку «Корзина» открывается страница basket.php. |
| R4.17 | Если корзина пуста, то на странице basket.php будет только ссылка «Корзина пуста. Добавьте что-нибудь» на главную страницу index.php. |
| R4.18 | Если корзина не пуста, то на странице будет таблица товаров с кнопками «Удалить» напротив каждой записи, информация об общей стоимости заказа и кнопка «Оформить заказ». |
| R4.19 | Если пользователь нажмет на кнопку «Удалить», то запись из корзины будет удалена |
| R4.20 | Если пользователь нажмет на кнопку «Оформить заказ», то откроется страница createOrder.php с полями для ввода адреса и телефона и кнопкой «Отправить». |
| R4.21 | Адрес должен содержать не менее семи символов и не более ста символов. |
| R4.22 | Если пользователь нажал на «Отправить» и адрес не соответствует требованиям «R4.20», то появится сообщение «Некорректный адрес». |
| R4.23 | Телефон должен состоять из семи цифр. |
| R4.24 | Если пользователь нажал на «Отправить» и телефон не соответствует требованиям «R4.22», то появится сообщение «Некорректный телефон». |
| R4.25 | Если пользователь заполнил все поля корректно, то заказ отправится на рассмотрение. |
| R4.26 | После нажатия на кнопку «Выход» пользователя перенаправляет на главную страницу index.php. |

## Неавторизованный пользователь

**Общие сведения**

Данную часть системы может видеть неавторизованный пользователь. Пользователь может выбрать одну из категорий товара, просмотреть товары, которые имеются в базе данных, после чего он может добавить (удалить из) их в корзину и оформить заказ. Неавторизованный пользователь может войти в систему или зарегистрироваться.

|  |  |
| --- | --- |
| R5.1 | На главной странице index.php находится шапка и ссылки по категориям: |
| R5.2 | После нажатия на одну из категорий открывается страница goods.php c таблицей товаров, строкой поиска и кнопкой «Поиск», ссылками сортировки «по названию» и «по цене» и шапкой как в index.php. |
| R5.3 | Поиск выдает все товары, чьё название содержит подстроку, которая соответствует строке в поле для поиска. |
| R5.4 | Если поиск ничего не вернул, то на месте таблицы будет надпись «Товаров с таким названием не найдено». |
| R5.5 | Если нажать на ссылку сортировки «по названию», то все строки в таблице сортируются по названию. |
| R5.6 | Если еще раз нажать на ссылку сортировки «по названию», то все строки в таблице сортируются по названию в обратном порядке. |
| R5.7 | Если нажать на ссылку сортировки «по цене», то все строки в таблице сортируются по цене. |
| R5.8 | Если еще раз нажать на ссылку сортировки «по цене», то все строки в таблице сортируются по цене в обратном порядке. |
| R5.9 | После нажатия на название товара в таблице открывается страница goodPage.php c названием, ценой, изображением, описанием товара и кнопкой «Добавить в корзину» |
| R5.10 | После нажатия на кнопку «Добавить в корзину» товар добавляется в корзину. |
| R5.11 | “Пекарь” в шапке – это ссылка на главную страницу index.php |
| R5.12 | После нажатия на ссылку «Корзина» открывается страница basket.php. |
| R5.13 | Если корзина пуста, то на странице basket.php будет только ссылка «Корзина пуста. Добавьте что-нибудь» на главную страницу index.php. |
| R5.14 | Если корзина не пуста, то на странице будет таблица товаров с кнопками «Удалить» напротив каждой записи, информация об общей стоимости заказа и кнопка «Оформить заказ». |
| R5.15 | Если пользователь нажмет на кнопку «Удалить», то запись из корзины будет удалена |
| R5.16 | Если пользователь нажмет на кнопку «Оформить заказ», то откроется страница createOrder.php с полями для ввода адреса и телефона и кнопкой «Отправить». |
| R5.17 | Адрес должен содержать не менее семи символов и не более ста символов. |
| R5.18 | Если пользователь нажал на «Отправить» и адрес не соответствует требованиям «R4.20», то появится сообщение «Некорректный адрес». |
| R5.19 | Телефон должен состоять из семи цифр. |
| R5.20 | Если пользователь нажал на «Отправить» и телефон не соответствует требованиям «R4.22», то появится сообщение «Некорректный телефон». |
| R5.21 | Если пользователь заполнил все поля корректно, то заказ отправится на рассмотрение. |
| R5.22 | После нажатия на ссылку «Регистрация» открывается страница registration.php |
| R5.23 | После нажатия на ссылку «Вход» открывается страница authentication.php |
| R5.24 | После нажатия на ссылку «Выход» пользователя перенаправляет на главную страницу index.php. |

# Проектирование программных модулей

Требуется разработать клиент-серверное приложение на языке C# с графическим интерфейсом для ОС MS Windows, использующее сокеты для сетевого взаимодействия.

Необходимо реализовать поиск файла, считывание атрибутов файла, изменение атрибутов файла, подсчет числа файлов в каталоге и получение размера каталога.

Клиент и сервер на транспортном уровне должны использовать протокол TCP. Серверной части приложения необходимо различать запросы от клиента, а клиенту необходимо «уметь» формировать запрос, поэтому нужно разработать собственный протокол взаимодействия клиента и сервера. Для этого можно создать перечисление, каждый элемент которого будет означать тот или иной запрос.

Для того, чтобы сервер мог работать с запросами от разных клиентов одновременно, он должен быть асинхронным либо многопоточным. В данном случае удобнее будет использовать пул потоков для взаимодействия с клиентами, размер которого задает системный администратор при запуске сервера.

Серверная часть должна работать с файлами, поэтому она будет активно взаимодействовать с файловой системой. Для работы с файловой системой на языке С# нужно использовать пространство имен System.IO.

Так как с сервером работает системный администратор, то серверную часть приложения можно сделать консольной.

Клиентская часть должна быть с удобным, интуитивно понятным и дружелюбным к пользователю графическим интерфейсом. Приложение может использовать Windows Form для графического интерфейса. Основная задача реализации клиентской части – обеспечение требуемого от приложения функционала взаимодействия с сервером и представление информации в удобном для пользователя виде.

# Реализация программных модулей

## Архитектура ПО и описание протокола взаимодействия клиента и сервера

Данное приложение использует архитектуру «клиент-сервер». На транспортном уровне используется протокол TCP. Для сетевого взаимодействия используются поточные сокеты (класс Socket из пространства имен System.Net.Sockets).

Для приема и отправки информации, а также для ее преобразования используется пользовательский класс Courier, который агрегирует сокет. Класс Courier включает в себя методы приема и отправки целых чисел и строк. Подробная UML-диаграмма данного класса представлена на рис. 3.1.1.



Рис. 3.1.1. – UML-диаграмма класса Courier

Для определения запросов сервера используется пользовательское перечисление ClientRequests, в котором каждый элемент означает конкретный запрос клиента серверу:

1. OpenFolderBrowser – дать список доступных устройств.
2. ShowSubFolders – дать список подпапок для данного пути.
3. SelectFolder – выбрать папку и отправить список всех доступных файлов, которые находятся в данной папке.
4. GetAttributes – дать атрибуты выбранного файла.
5. SetAttributes – установить новые атрибуты для данного файла.
6. CloseConnection – завершить соединение

На рис. 3.1.2 приведена UML-диаграмма данного перечисления.



Рис. 3.1.2. – UML-диаграмма перечисления ClientRequests

Также в приложении используется утилитный класс PathUtility, который облегчает работу с путями к файлам и папкам. Его UML-диаграмма представлена на рис. 3.1.3.



Рис. 3.1.3. – UML-диаграмма класса PathUtility

## Серверная часть

Сервер использует список потоков для взаимодействия с множеством клиентов одновременно. Максимально допустимое количество одновременно обслуживаемых клиентов задается системным администратором при запуске программы. Для каждого подключившегося клиента сервер создает поток и добавляет этот поток в список.

На сервере, помимо основного потока и списка потоков, работает еще один поток, который наблюдает за состоянием потоков в списке. Если поток мертв – наблюдающий поток удаляет его из списка потоков и, таким образом, освобождается место для подключения новых клиентов. Наблюдающий поток использует метод MonitorThreads() пользовательского класса ThreadMonitor, UML-диаграмма которого представлена на рис. 3.2.1.



Рис. 3.2.1. – UML-диаграмма класса ThreadMonitor

Основная логика взаимодействия сервера с отдельным клиентом содержится в методе RunService() пользовательского класса ClientObject. Этот метод используется в потоках, созданных для клиента. На рис. 3.2.2 изображена UML-диаграмма класса ClientObject. Данный класс содержит все необходимые методы для взаимодействия с клиентом.



Рис 3.2.2. – UML-диаграмма класса ClientObject

За работу с устройствами, папками и файлами отвечает пользовательский класс FileManager. С помощью данного класса можно получить список файлов, папок и доступных устройств. Его UML-диаграмма изображена на рис. 3.2.3. Также данный класс имеет методы для нахождения размера папки и представлении его в удобном (строковом) виде.



Рис. 3.2.3. – UML-диаграмма класса FileManager

## Клиентская часть

Клиентская часть использует графический интерфейс для взаимодействия с пользователями. На главной форме (рис. 3.3.1) находится listBox, в который выводится список файлов, поисковая строка и другие компоненты.

Рис. 3.3.1. – Схематичное изображение главной формы

Для просмотра и выбора папки используется еще одна форма (рис. 3.3.2).



Рис 3.3.2. – Схематичное изображение формы для выбора папки

Изображение формы для просмотра и изменения атрибутов выбранного файла приведено на рис. 3.3.3.



Рис 3.3.3. – Схематичное изображение формы для изменения атрибутов файла

При нажатии на какую-либо из кнопок, при котором должно происходить сетевое взаимодействие, выполняется сначала проверка соединения. Проверка соединения выполняется так: пытаемся серверу отправить массив байтов, состоящий из 0 байт, если удалось отправить, значит с соединением все в порядке. Если произошла ошибка, то программа уведомит об этом пользователя.

# Проектирование модульных тестов

## Построение ориентированных графов Мак-кейба

Сервер можно запустить через командную строку (или с помощью Windows PowerShell) без параметров (рис. 4.1.1) и с одним параметром (рис. 4.1.2), который означает максимальное количество одновременно обслуживаемых клиентов.

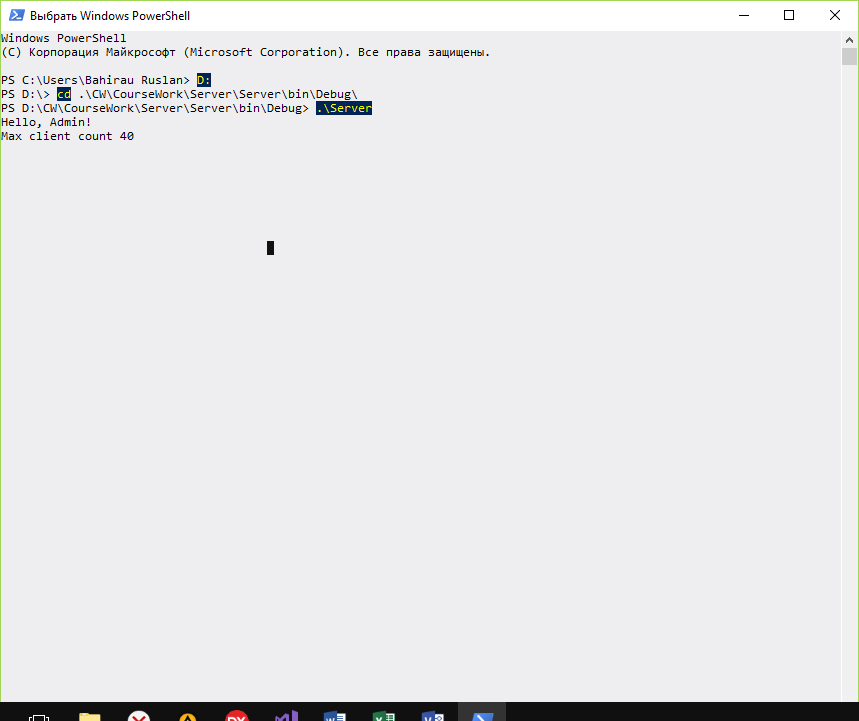


Рис. 4.1.1 – Запуск сервера без параметров

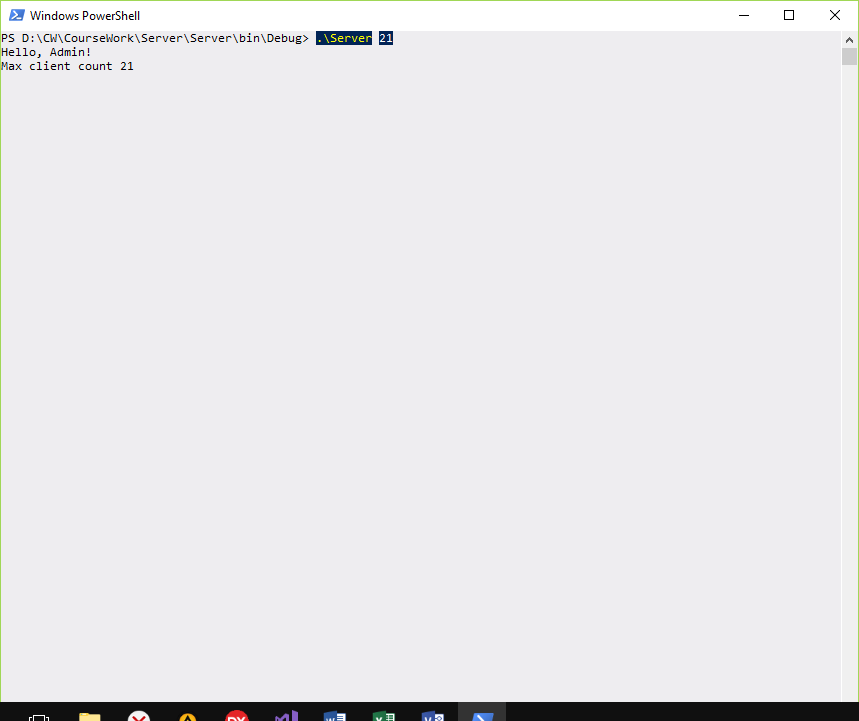


Рис. 4.1.2. – Запуск сервера с параметром, указывающим максимальное количество одновременно обслуживаемых клиентов

Теперь в командной строке будет отображаться информация о новых клиентах, о прекращении соединения и об ошибках (рис. 4.1.3).

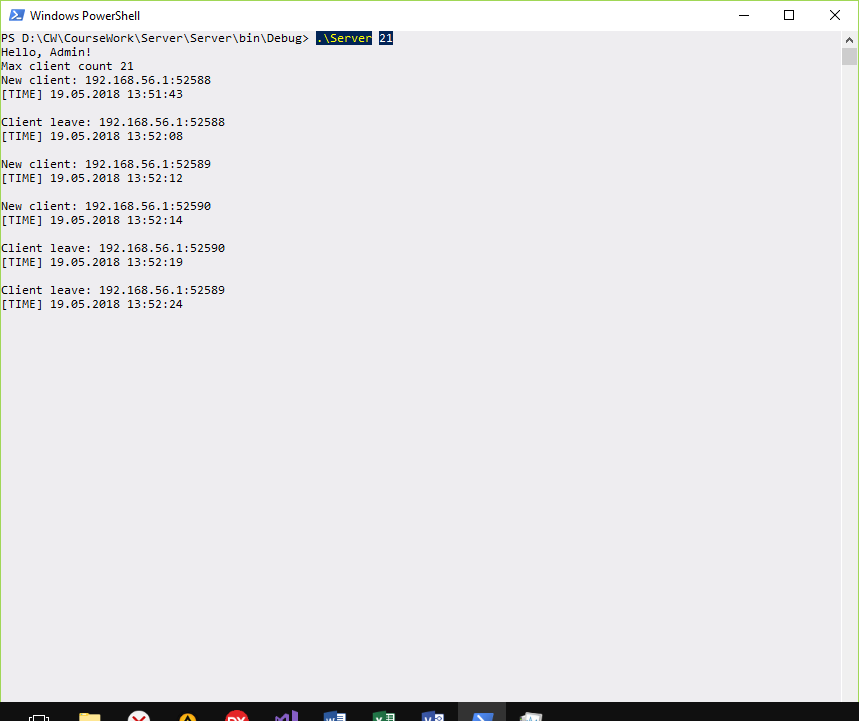


Рис. 4.1.3 – Работа сервера с клиентами

## Формирование проходов

Запустить программу можно просто выполнив файл Client.exe. Появится главное окно приложения (рис. 4.2.1).

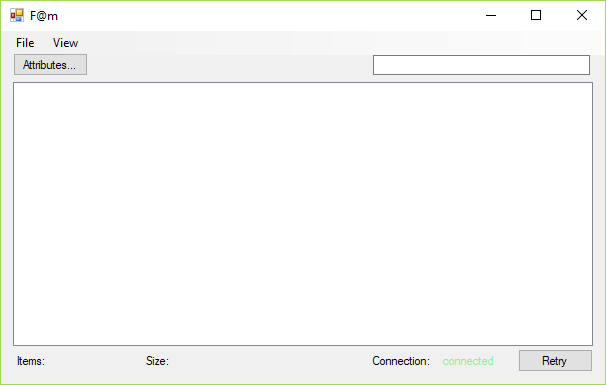


Рис. 4.2.1. – Главное окно приложения

Снизу есть три показателя: «Items», «Size» и «Connection». «Items» показывает количество файлов, доступных пользователю на главной форме. «Size» показывает размер текущей папки. «Connection» показывает состояние соединения с сервером.

Выбрав «File -> Open» откроется окно для выбора папки (рис. 4.2.2). Щелкните два раза по папке, к которой хотите перейти. Нажмите «Select Folder» чтобы выбрать папку. После этих действий на главной форме появится список файлов (рис. 4.2.3).

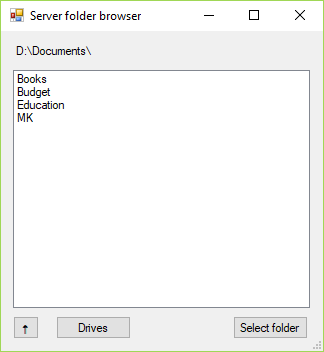


Рис. 4.2.2. – Выбор папки

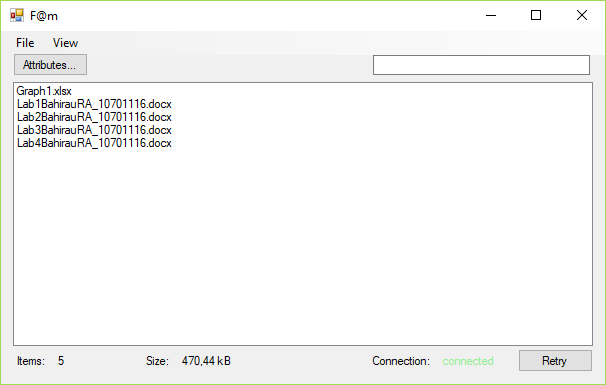


Рис. 4.2.3. – Список файлов на форме

Теперь, если вы хотите найти имя файла с определенной подстрокой, просто введите ее в поле для поиска (рис. 4.2.4).

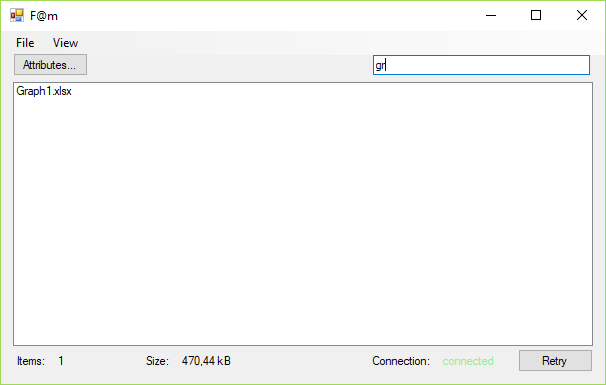


Рис. 4.2.4. – Поиск файлов из списка с подстрокой «gr»

Выберите файл из списка, щелкнув по его имени, и нажмите кнопку «Attributes…». Откроется окно для просмотра и изменения атрибутов файла (рис. 4.2.5).

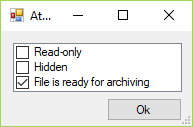


Рис. 4.2.5. – Окно атрибутов файла

Просто выберите и измените нужные атрибуты и нажмите на кнопку «Ok». Готово, теперь атрибуты изменены.

Если нет соединения с сервером или оно потеряно, то появится сообщение (рис 4.2.6).

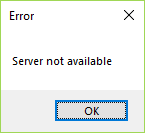


Рис. 4.2.6. – Сообщение о потере соединения

Для попытки подключения к серверу просто нажмите на кнопку «Retry». Если соединение возобновилось, то индикатор «Connection» на главной форме покажет текст салатового цвета.

# Реализация модульного тестирования

Для попытки подключения к серверу просто нажмите на кнопку «Retry». Если соединение возобновилось, то индикатор «Connection» на главной форме покажет текст салатового цвета.

## Реализация автоматизированных скриптов

## Анализ результатов тестов

## Отладка тестов

# Заключение

При выполнении данного курсового проекта мною были освоены приемы работы с поточными сокетами и разработан простой протокол взаимодействия между клиентом и сервером. Я ознакомился с классами пространств имен System.Net, System.Net.Sockets, System.IO, System.Threading и т.д. На практике закрепил умение разрабатывать многопоточные приложения и приложения, использующие графический интерфейс. Также я узнал больше об атрибутах файла/папки и как с ними работать.

Результатом выполнения данного курсового проекта стала реализация на практике умений разрабатывать многопоточные клиент-серверные приложения, использующие поточные сокеты для сетевого взаимодействия.

# Список использованных источников

1. Олифер, В. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов./Олифер В., Олифер Н. – 5-е изд. – СПб.: Питер, 2016. – 992с.
2. Пространство имен System.IO. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/system.io(v=vs.110).aspx.
3. System.Net – пространства имен. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/gg145039(v=vs.110).aspx.
4. Пространство имен System.Net.Sockets. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/system.net.sockets(v=vs.110).aspx.

# ПРИЛОЖЕНИЕ А

## Листинг исходных кодов

model/entity/CartRecord.php

**<?php  
  
require\_once "Product.php"**;  
  
**class** CartRecord  
{  
 **private $product**;  
 **private $count**;  
  
 **public function** \_\_construct($product, $count)  
 {  
 $this->setProduct($product);  
 $this->setCount($count);  
 }  
  
 **public function** getProduct()  
 {  
 **return** $this->**product**;  
 }  
  
 **public function** setProduct($product)  
 {  
 **if** ($product **instanceof** Product)  
 {  
 $this->**product** = $product;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"product"**);  
 }  
 }  
  
 **public function** getCount()  
 {  
 **return** $this->**count**;  
 }  
  
 **public function** setCount($count)  
 {  
 **if** (*is\_numeric*($count)  
 && $count >= 1 && $count <= 256)  
 {  
 $this->**count** = $count;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"count"**);  
 }  
 }  
}

model/entity/Category.php

**<?php  
  
require\_once "Identificational.php"**;  
  
**class** Category **extends** Identificational  
{  
 **private $name**;  
  
 **public function** \_\_construct($id, $name)  
 {  
 **parent**::*\_\_construct*($id);  
 $this->setName($name);  
 }  
  
  
 **public function** getName()  
 {  
 **return** $this->**name**;  
 }  
  
 **public function** setName($name)  
 {  
 **if** (*strlen*($name) >= 2 && *strlen*($name) <= 64)  
 {  
 $this->**name** = $name;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"name"**);  
 }  
 }  
}

model/entity/HistoryRecord.php

**<?php  
  
require\_once "OrderRecord.php"**;  
  
**class** HistoryRecord **extends** OrderRecord  
{  
 **private $confirmDate**;  
  
 **public function** \_\_construct($id, $user, $product, $count, $address,  
 $phone, $orderDate, $confirmDate)  
 {  
 **parent**::*\_\_construct*($id, $user, $product, $count, $address, $phone,  
 $orderDate);  
 $this->setConfirmDate($confirmDate);  
 }  
  
 **public function** getConfirmDate()  
 {  
 **return** $this->**confirmDate**;  
 }  
  
 **public function** setConfirmDate($confirmDate)  
 {  
 **if** (*date\_create\_from\_format*(**"Y-m-d"**, $confirmDate)  
 && $confirmDate <= *date*(**"Y-m-d"**)  
 && $confirmDate >= $this->getOrderDate())  
 {  
 $this->**confirmDate** = $confirmDate;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"confirmDate"**);  
 }  
 }  
}

model/entity/Identificational.php

**<?php  
  
class** Identificational  
{  
 **private $id**;  
  
 **protected function** \_\_construct($id)  
 {  
 $this->setId($id);  
 }  
  
 **public function** getId()  
 {  
 **return** $this->**id**;  
 }  
  
 **public function** setId($id)  
 {  
 **if** (*is\_numeric*($id) && $id >= 0 && $id <= *pow*(2, 32))  
 {  
 $this->**id** = $id;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"id"**);  
 }  
 }  
}

model/entity/OrderRecord.php

**<?php  
  
require\_once "User.php"**;  
**require\_once "Product.php"**;  
  
**class** OrderRecord **extends** Identificational  
{  
 **private $user**;  
 **private $product**;  
 **private $count**;  
 **private $address**;  
 **private $phone**;  
 **private $orderDate**;  
  
 **public function** \_\_construct($id, $user, $product, $count, $address,  
 $phone, $orderDate)  
 {  
 **parent**::*\_\_construct*($id);  
 $this->setUser($user);  
 $this->setProduct($product);  
 $this->setCount($count);  
 $this->setAddress($address);  
 $this->setPhone($phone);  
 $this->setOrderDate($orderDate);  
 }  
  
 **public function** getUser()  
 {  
 **return** $this->**user**;  
 }  
  
 **public function** setUser($user)  
 {  
 $this->**user** = $user;  
 }  
  
 **public function** getProduct()  
 {  
 **return** $this->**product**;  
 }  
  
 **public function** setProduct($product)  
 {  
 **if** ($product **instanceof** Product)  
 {  
 $this->**product** = $product;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"product"**);  
 }  
 }  
  
 **public function** getCount()  
 {  
 **return** $this->**count**;  
 }  
  
 **public function** setCount($count)  
 {  
 **if** (*is\_numeric*($count)  
 && $count >= 1 && $count <= 256)  
 {  
 $this->**count** = $count;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"count"**);  
 }  
 }  
  
 **public function** getAddress()  
 {  
 **return** $this->**address**;  
 }  
  
 **public function** setAddress($address)  
 {  
 **if** (*strlen*($address) >= 7  
 && *strlen*($address) <= 100)  
 {  
 $this->**address** = $address;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"address"**);  
 }  
 }  
  
 **public function** getPhone()  
 {  
 **return** $this->**phone**;  
 }  
  
 **public function** setPhone($phone)  
 {  
 **if** (*strlen*($phone) == 7 && *is\_numeric*($phone))  
 {  
 $this->**phone** = $phone;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"phone"**);  
 }  
 }  
  
 **public function** getOrderDate()  
 {  
 **return** $this->**orderDate**;  
 }  
  
 **public function** setOrderDate($orderDate)  
 {  
 **if** (*date\_create\_from\_format*(**"Y-m-d"**, $orderDate)  
 && $orderDate <= *date*(**"Y-m-d"**))  
 {  
 $this->**orderDate** = $orderDate;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"orderDate"**);  
 }  
 }  
}

model/entity/Product.php

**<?php  
  
require\_once "Category.php"**;  
  
**class** Product **extends** Identificational  
{  
 **private $name**;  
 **private $category**;  
 **private $price**;  
 **private $description**;  
  
 **public function** \_\_construct($id, $name, $category, $price, $description)  
 {  
 **parent**::*\_\_construct*($id);  
 $this->setName($name);  
 $this->setCategory($category);  
 $this->setPrice($price);  
 $this->setDescription($description);  
 }  
  
 **public function** getName()  
 {  
 **return** $this->**name**;  
 }  
  
 **public function** setName($name)  
 {  
 **if** (*strlen*($name) >= 5 && *strlen*($name) <= 128)  
 {  
 $this->**name** = $name;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"name"**);  
 }  
 }  
  
 **public function** getCategory()  
 {  
 **return** $this->**category**;  
 }  
  
 **public function** setCategory($category)  
 {  
 **if** ($category **instanceof** Category)  
 {  
 $this->**category** = $category;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"category"**);  
 }  
 }  
  
 **public function** getPrice()  
 {  
 **return** $this->**price**;  
 }  
  
 **public function** setPrice($price)  
 {  
 **if** (*is\_numeric*($price) && $price >= 0  
 && $price <= *pow*(2, 32))  
 {  
 $this->**price** = $price;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"price"**);  
 }  
 }  
  
 **public function** getDescription()  
 {  
 **return** $this->**description**;  
 }  
  
 **public function** setDescription($description)  
 {  
 **if** (*strlen*($description) <= 10000)  
 {  
 $this->**description** = $description;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"description"**);  
 }  
 }  
}

model/entity/User.php

**<?php  
  
require\_once "Identificational.php"**;  
  
**class** User **extends** Identificational  
{  
 **private $login**;  
 **private $email**;  
 **private $password**;  
  
 **public function** \_\_construct($id, $login, $email, $password)  
 {  
 **parent**::*\_\_construct*($id);  
 $this->setLogin($login);  
 $this->setEmail($email);  
 $this->setPassword($password);  
 }  
  
 **public function** getLogin()  
 {  
 **return** $this->**login**;  
 }  
  
 **public function** setLogin($login)  
 {  
 **if** (*preg\_match*(**'/^[a-z0-9\_-]{5,64}$/'**, $login))  
 {  
 $this->**login** = $login;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"login"**);  
 }  
 }  
  
 **public function** getEmail()  
 {  
 **return** $this->**email**;  
 }  
  
 **public function** setEmail($email)  
 {  
 **if** (*preg\_match*(**'/^([a-z0-9\_\.-]+)@([a-z0-9\_\.-]+)\.([a-z\.]{2,6})$/'**,  
 $email) && *strlen*($email) <= 128)  
 {  
 $this->**email** = $email;  
 }  
 **else** {  
 **throw new** InvalidArgumentException(**"email"**);  
 }  
 }  
  
 **public function** getPassword()  
 {  
 **return** $this->**password**;  
 }  
  
 **public function** setPassword($password)  
 {  
 $this->**password** = $password;  
 }  
}

model/logic/AuthenticationVerifyer.php

**<?php  
  
require\_once *\_\_DIR\_\_*** . **'\..\util\dao\UserDao.php'**;  
  
**class** AuthenticationVerifyer  
{  
 **private $error\_msgs** = **array**();  
  
 **public function** \_\_construct(**array** $error\_msgs)  
 {  
 $this->**error\_msgs** = $error\_msgs;  
 }  
  
 **public function** authenticationErrors($login, $password, $mysqli) {  
 **if** (!$this->isCorrectLogin($login)) {  
 **return** $this->**error\_msgs**[**'incorrect\_login'**];  
 } **else if** (!$this->isCorrectPassword($password)) {  
 **return** $this->**error\_msgs**[**'incorrect\_password'**];  
 } **else if** (!$this->isTrueUser($login, $password, $mysqli)) {  
 **return** $this->**error\_msgs**[**'not\_user'**];  
 } **else** {  
 **return false**;  
 }  
 }  
  
 **public function** isTrueUser($login, $password, $mysqli) {  
 **if** ($this->isExistUser($login, $mysqli)) {  
 $dao = **new** UserDao($mysqli);  
 $user = $dao->getBy(**'login'**, $login)[0];  
 $hash = $user->getPassword();  
 **return** *password\_verify*($password, $hash);  
 }  
 **return false**;  
 }  
  
 **public function** isExistUser($login, $mysqli) {  
 $userDao = **new** UserDao($mysqli);  
 $users = $userDao->getBy(**'login'**, $login);  
 **return** *count*($users) != 0;  
 }  
  
 **public function** isCorrectLogin($login) {  
 **return** $this->isSameFormat(**'/^[a-z0-9\_-]{5,64}$/'**, $login);  
 }  
  
 **public function** isCorrectPassword($password) {  
 **return** (*strlen*($password) >= 5) && (*strlen*($password) <= 32);  
 }  
  
 **public function** isSameFormat($regexp, $string) {  
 **return** (bool) *preg\_match*($regexp, $string);  
 }  
}

model/logic/Authenticator.php

**<?php  
  
class** Authenticator  
{  
 **public function** authenticate($login, $error)  
 {  
 **if** (!$error) {  
 $\_SESSION[**"logged\_user"**] = $login;  
 }  
 }  
}

model/logic/Cashier.php

**<?php  
  
require\_once *\_\_DIR\_\_*** . **'\..\entity\CartRecord.php'**;  
  
**class** Cashier  
{  
 **private static** *$cashier*;  
  
 **private function** \_\_construct(){}  
  
 **public static function** getCashier()  
 {  
 **if** (**self**::*$cashier* === **null**)  
 {  
 **self**::*$cashier* = **new self**();  
 }  
 **return self**::*$cashier*;  
 }  
   
 **public function** getTotalPrice($cart)  
 {  
 $totalPrice = 0;  
 **if** (!*is\_array*($cart))  
 {  
 **throw new** InvalidArgumentException(**"not array"**);  
 }  
 **foreach** ($cart **as** $record)  
 {  
 **if** ($record **instanceof** CartRecord)  
 {  
 $totalPrice += $record->getProduct()->getPrice()  
 \* $record->getCount();  
 }  
 }  
 **return** $totalPrice;  
 }  
}

model/logic/CategoryVerifyer.php

**<?php  
  
require\_once *\_\_DIR\_\_*** . **'\..\util\dao\CategoryDao.php'**;  
  
**class** CategoryVerifyer  
{  
  
 **private $error\_msgs** = **array**();  
  
 **public function** \_\_construct(**array** $error\_msgs)  
 {  
 $this->**error\_msgs** = $error\_msgs;  
 }  
  
 **public function** categoryErrors($name, $mysqli) {  
 **if** (!$this->isCorrectName($name)) {  
 **return** $this->**error\_msgs**[**"incorrect\_name"**];  
 } **else if** ($this->isExistCategoryName($name, $mysqli)) {  
 **return** $this->**error\_msgs**[**"double\_name"**];  
 } **else** {  
 **return false**;  
 }  
 }  
  
 **public function** isExistCategoryName($name, $mysqli)  
 {  
 $dao = **new** CategoryDao($mysqli);  
 $categories = $dao->getBy(**'name'**, $name);  
 **return** *count*($categories) != 0;  
 }  
  
 **public function** isCorrectName($name)  
 {  
 **return** *strlen*($name) >= 2 && *strlen*($name) <= 64;  
 }  
}

model/logic/FaceControl.php

**<?php  
  
require\_once *\_\_DIR\_\_*** . **'\..\util\session.php'**;   
  
**class** FaceControl  
{  
 **private static** *$faceControl*;  
  
 **private function** \_\_construct(){}  
  
 **public static function** getFaceControl()  
 {  
 **if** (**self**::*$faceControl* === **null**)  
 {  
 **self**::*$faceControl* = **new self**();  
 }  
 **return self**::*$faceControl*;  
 }  
  
 **public function** isAdmin()  
 {  
 **return isset**($\_SESSION[**'logged\_user'**])  
 && $\_SESSION[**'logged\_user'**] == **'admin'**;  
 }  
  
 **public function** getOneOf($if\_guest, $if\_user, $if\_admin=**null**)  
 {  
 **if** (!**isset**($\_SESSION[**'logged\_user'**]))  
 {  
 **return** $if\_guest;  
 }  
 **else if** ($\_SESSION[**'logged\_user'**] == **'admin'**)  
 {  
 **return** $if\_admin;  
 }  
 **else** {  
 **return** $if\_user;  
 }  
 }  
}

model/logic/OrderVerifyer.php

**<?php  
  
class** OrderVerifyer  
{  
  
 **private $error\_msgs** = **array**();  
  
 **public function** \_\_construct(**array** $error\_msgs)  
 {  
 $this->**error\_msgs** = $error\_msgs;  
 }  
  
 **public function** orderErrors($address, $phone) {  
 **if** (!$this->isCorrectAddress($address)) {  
 **return** $this->**error\_msgs**[**"incorrect\_address"**];  
 } **else if** (!$this->isCorrectPhone($phone)) {  
 **return** $this->**error\_msgs**[**"incorrect\_phone"**];  
 } **else** {  
 **return false**;  
 }  
 }  
  
 **public function** isCorrectAddress($name)  
 {  
 **return** *strlen*($name) >= 7 && *strlen*($name) <= 100;  
 }  
  
 **public function** isCorrectPhone($phone)  
 {  
 **return** *strlen*($phone) == 7 && *is\_numeric*($phone);  
 }  
  
 **public function** isCorrectCount($count)  
 {  
 **return** *is\_numeric*($count) && $count >= 1 && $count <= 256;  
 }  
}

model/logic/ProductVerifyer.php

**<?php  
  
require\_once *\_\_DIR\_\_*** . **'\..\util\dao\ProductDao.php'**;   
  
**class** ProductVerifyer  
{  
  
 **private $error\_msgs** = **array**();  
  
 **public function** \_\_construct(**array** $error\_msgs)  
 {  
 $this->**error\_msgs** = $error\_msgs;  
 }  
  
 **public function** productErrors($name, $price, $description, $mysqli) {  
 **if** (!$this->isCorrectName($name)) {  
 **return** $this->**error\_msgs**[**"incorrect\_name"**];  
 } **else if** (!$this->isCorrectPrice($price)) {  
 **return** $this->**error\_msgs**[**"incorrect\_price"**];  
 } **else if** (!$this->isCorrectDescription($description)) {  
 **return** $this->**error\_msgs**[**"big\_description"**];  
 } **else if** ($this->isExistName($name, $mysqli)) {  
 **return** $this->**error\_msgs**[**"double\_name"**];  
 } **else** {  
 **return false**;  
 }  
 }  
  
 **public function** isExistName($name, $mysqli)  
 {  
 $dao = **new** ProductDao($mysqli);  
 $categories = $dao->getBy(**'name'**, $name);  
 **return** *count*($categories) != 0;  
 }  
  
 **public function** isCorrectName($name)  
 {  
 **return** *strlen*($name) >= 5 && *strlen*($name) <= 128;  
 }  
  
 **public function** isCorrectPrice($price)  
 {  
 **return** $price >= 0 && $price <= 2147483648;  
 }  
  
 **public function** isCorrectDescription($description)  
 {  
 **return** *strlen*($description) <= 10000;  
 }  
}

model/logic/RegistrationVerifyer.php

**<?php  
  
require\_once *\_\_DIR\_\_*** . **'\..\util\dao\UserDao.php'**;  
  
**class** RegistrationVerifyer  
{  
 **private $error\_msgs** = **array**();  
  
 **public function** \_\_construct(**array** $error\_msgs)  
 {  
 $this->**error\_msgs** = $error\_msgs;  
 }  
  
 **public function** registrationErrors($login, $email, $password, $password2, $mysqli) {  
 **if** (!**self**::*isCorrectLogin*($login)) {  
 **return** $this->**error\_msgs**[**"incorrect\_login"**];  
 } **else if** (!$this->isCorrectPassword($password)) {  
 **return** $this->**error\_msgs**[**"incorrect\_password"**];  
 } **else if** (!$this->isCorrectEmail($email)) {  
 **return** $this->**error\_msgs**[**"incorrect\_email"**];  
 } **else if** ($this->isExistUser($login, $mysqli)) {  
 **return** $this->**error\_msgs**[**"double\_login"**];  
 } **else if** ($this->isExistEmail($email, $mysqli)) {  
 **return** $this->**error\_msgs**[**"double\_email"**];  
 } **else if** ($password != $password2) {  
 **return** $this->**error\_msgs**[**"incorrect\_repeat"**];  
 } **else** {  
 **return false**;  
 }  
 }  
  
 **public function** isExistUser($login, $mysqli) {  
 $userDao = **new** UserDao($mysqli);  
 $users = $userDao->getBy(**'login'**, $login);  
 **return** *count*($users) != 0;  
 }  
  
 **public function** isExistEmail($email, $mysqli) {  
 $userDao = **new** UserDao($mysqli);  
 $users = $userDao->getBy(**'email'**, $email);  
 **return** *count*($users) != 0;  
 }  
  
 **public function** isCorrectLogin($login) {  
 **return self**::*isSameFormat*(**'/^[a-z0-9\_-]{5,64}$/'**, $login);  
 }  
  
 **public function** isCorrectPassword($password) {  
 **return** (*strlen*($password) >= 5) && (*strlen*($password) <= 32);  
 }  
  
 **public function** isCorrectEmail($email) {  
 **return self**::*isSameFormat*(**'/^([a-z0-9\_\.-]+)@([a-z0-9\_\.-]+)\.([a-z\.]{2,6})$/'**,  
 $email) && *strlen*($email) <= 128;  
 }  
  
 **public function** isSameFormat($regexp, $string) {  
 **return** (bool) *preg\_match*($regexp, $string);  
 }  
}

model/logic/Registrator.php

**<?php  
  
require\_once *\_\_DIR\_\_*** . **'\..\util\dao\UserDao.php'**;  
  
**class** Registrator  
{  
 **public function** registrateUser($login, $email, $password, $errors, $db)  
 {  
 **if** (!$errors) {  
 $hash = *password\_hash*($password, ***PASSWORD\_DEFAULT***);  
 $dao = **new** UserDao($db);  
 $dao->add(**new** User(0, $login, $email, $hash));  
 }  
 }  
}

model/logic/URIResolver.php

**<?php  
  
class** URIResolver  
{  
 **private static** *$uriResolver*;  
  
 **private function** \_\_construct()  
 {  
 }  
  
 **public static function** getURIResolver()  
 {  
 **if** (**self**::*$uriResolver* === **null**) {  
 **self**::*$uriResolver* = **new self**();  
 }  
 **return self**::*$uriResolver*;  
 }  
  
 **public function** getValue($name)  
 {  
 **if** ($this->hasGET($name)) {  
 **return** $\_GET[$name];  
 }  
 **return false**;  
 }  
  
 **public function** getPOSTValue($name)  
 {  
 **if** ($this->hasPOST($name)) {  
 **return** $\_POST[$name];  
 }  
 **return false**;  
 }  
  
 **public function** hasGET($name)  
 {  
 **return isset**($\_GET[$name]);  
 }  
  
 **public function** hasPOST($name)  
 {  
 **return isset**($\_POST[$name]);  
 }  
  
 **public function** setToURI($uri, $name, $value)  
 {  
 **if** (*strripos*($uri, **'?'**) !== **false**)  
 {  
 $valuess = *substr*($uri, *strripos*($uri, **'?'**) + 1);  
 $values = *explode*(**'&'**, $valuess);  
 **foreach** ($values **as** $val) {  
 **if** (*strripos*($val, $name . **'='**) === 0) {  
 $replace = $name . **'='** . $value;  
 **return** *str\_replace*($val, $replace, $uri);  
 }  
 }  
 **return** $uri . **'&'** . $name . **'='** . $value;  
 }  
 **else** {  
 **return** $uri . **'?'** . $name . **'='** . $value;  
 }  
 }  
  
 **public function** clearURI($uri)  
 {  
 **if** (*strripos*($uri, **'?'**) !== **false**) {  
 **return** *substr*($uri, 0, *strripos*($uri, **'?'**));  
 }  
 **return** $uri;  
 }  
  
 **public function** getOnlyValues($uri)  
 {  
 **return** *substr*($uri, *strripos*($uri, **'?'**));  
 }  
  
 **public function** unsetFromURI($uri, $name)  
 {  
 $valuess = *substr*($uri, *strripos*($uri, **'?'**) + 1);  
 $values = *explode*(**'&'**, $valuess);  
 $valname = **""**;  
 **foreach** ($values **as** $val) {  
 **if** (*strripos*($val, $name . **'='**) === 0) {  
 $valname = $val;  
 }  
 }  
 **if** (*strripos*($valuess, $valname) === 0 && *strripos*($valuess, **"&"**)) {  
 **return** *str\_replace*($valname . **'&'**, **''**, $uri);  
 } **else if** (*strripos*($valuess, $valname) === 0) {  
 **return** *str\_replace*(**'?'** . $valname, **''**, $uri);  
 } **else** {  
 **return** *str\_replace*(**'&'** . $valname, **''**, $uri);  
 }  
 }  
}

# ПРИЛОЖЕНИЕ Б

## Листинг автоматизированных скриптов

MainServer.cs

using System;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.Collections.Generic;

using MyAddedTypes;

namespace Server

{

class MainServer

{

private static int PORT = 5555;

private static int MAX\_CLIENT\_COUNT = 40;

static void Main(string[] args)

{

try

{

int clientCount = getClientCount(args);

Console.WriteLine("Hello, Admin!\nMax client count {0}",

clientCount);

Socket listener = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

listener.Bind(new IPEndPoint(IPAddress.Any, PORT));

listener.Listen(clientCount);

Run(listener, clientCount);

listener.Shutdown(SocketShutdown.Both);

listener.Close();

}

finally

{

Console.Write(@"Press 'Enter' to exit: ");

Console.ReadLine();

}

}

private static void Run(Socket listener, int clientCount)

{

List<Thread> threads = new List<Thread>();

Thread monitor = new Thread(new ThreadStart(new ThreadMonitor(threads).MonitorThreads));

monitor.Start();

while (true)

{

if (threads.Count < clientCount)

{

Socket clientSocket = listener.Accept();

Console.WriteLine("New client: {0}", clientSocket.RemoteEndPoint);

Console.WriteLine("[TIME] {0}\n", DateTime.Now);

monitor.Suspend();

threads.Add(new Thread(new ThreadStart(

new ClientObject(clientSocket).RunService)));

threads[threads.Count - 1].Start();

monitor.Resume();

}

}

}

private static int getClientCount(string[] args)

{

int clientCount = MAX\_CLIENT\_COUNT;

if (args.Length > 0)

{

int.TryParse(args[0], out clientCount);

}

return clientCount;

}

}

}

ClientObject.cs

using System;

using System.IO;

using System.Net.Sockets;

using MyAddedTypes;

namespace Server

{

class ClientObject

{

private Courier courier;

public ClientObject(Socket socket)

{

courier = new Courier(socket);

}

public void RunService()

{

while (true)

{

try

{

ClientRequests request = WaitRequest();

switch (request)

{

case ClientRequests.OpenFolderBrowser:

SendOpenFolderBrowser();

break;

case ClientRequests.ShowSubFolders:

SendSubFolders(courier.ReceiveString(),

bool.Parse(courier.ReceiveString()));

break;

case ClientRequests.SelectFolder:

SendFiles(courier.ReceiveString(),

bool.Parse(courier.ReceiveString()));

break;

case ClientRequests.GetAttributes:

SendAttributes(courier.ReceiveString());

break;

case ClientRequests.SetAttributes:

SetAttributes(courier.ReceiveString(),

courier.ReceiveInteger());

break;

case ClientRequests.CloseConnection:

default:

ViewInfoLeave();

courier.Leave();

return;

}

}

catch (SocketException e)

{

Console.WriteLine("Problem: \n" + e);

return;

}

}

}

private void SendOpenFolderBrowser()

{

string[] drives = FileManager.GetDrives();

courier.SendInteger(drives.Length);

foreach (string drive in drives)

{

courier.SendString(drive);

}

}

private void SendFiles(string path, bool hidden)

{

path = PathUtility.ClosePath(path);

string[] files = hidden ? FileManager.GetFiles(path)

: FileManager.GetFilesWithoutHidden(path);

courier.SendInteger(files.Length);

foreach (string file in files)

{

courier.SendString(file);

}

courier.SendString(FileManager.FormatSize(path, 2));

}

private void SendAttributes(string file)

{

courier.SendInteger((int)File.GetAttributes(file));

}

private void SetAttributes(string file, int attributes)

{

File.SetAttributes(file, (FileAttributes)attributes);

}

private void SendSubFolders(string path, bool hidden)

{

path = PathUtility.ClosePath(path);

string[] directories = hidden ? FileManager.GetDirectories(path)

: FileManager.GetDirectoriesWithoutHidden(path);

courier.SendInteger(directories.Length);

foreach (string directory in directories)

{

courier.SendString(PathUtility.GetNameWithoutParent(directory));

}

}

private ClientRequests WaitRequest()

{

return (ClientRequests) courier.ReceiveInteger();

}

private void ViewInfoLeave()

{

Console.WriteLine("Client leave: {0}", courier.GetSocket().RemoteEndPoint);

Console.WriteLine("[TIME] {0}\n", DateTime.Now);

}

}

}

FileManager.cs

using System;

using System.Collections.Generic;

using System.IO;

namespace MyAddedTypes

{

class FileManager

{

public static string[] GetFilesWithoutHidden(string path)

{

return PathsWithout(GetFiles(path),

FileAttributes.Hidden);

}

public static string[] GetDirectoriesWithoutHidden(string path)

{

return PathsWithout(GetDirectories(path),

FileAttributes.Hidden);

}

public static string[] GetFiles(string path)

{

string[] paths;

try

{

paths = Directory.GetFiles(path);

}

catch (UnauthorizedAccessException)

{

paths = new string[0];

}

return PathsWithout(paths, FileAttributes.System);

}

public static string[] GetDirectories(string path)

{

string[] paths;

try

{

paths = Directory.GetDirectories(path);

}

catch (UnauthorizedAccessException)

{

paths = new string[0];

}

return PathsWithout(paths, FileAttributes.System);

}

public static string[] GetDrives()

{

DriveInfo[] drives = DriveInfo.GetDrives();

List<string> listDrives = new List<string>();

foreach (DriveInfo drive in drives)

{

try

{

Directory.GetFiles(drive.ToString());

listDrives.Add(drive.ToString());

}

catch (IOException) { }

catch (UnauthorizedAccessException) { }

}

return listDrives.ToArray();

}

public static string FormatSize(string path, int nums)

{

int uSize = 0;

double fSize = DirSize(path);

string[] unit = { "b", "kB", "MB", "GB", "TB" };

while (fSize >= 1024 && uSize < unit.Length)

{

fSize /= 1024;

uSize++;

}

return string.Format("{0} {1}",

Math.Round(fSize, nums), unit[uSize]);

}

private static string[] PathsWithout(string[] paths,

FileAttributes fileAttributes)

{

List<string> listWithoutSystem = new List<string>();

foreach (string path in paths)

{

if (!File.GetAttributes(path).HasFlag(fileAttributes))

{

listWithoutSystem.Add(path);

}

}

return listWithoutSystem.ToArray();

}

private static long DirSize(string path)

{

long size = 0;

string[] files = GetFiles(path);

foreach (string file in files)

{

size += (new FileInfo(file)).Length;

}

string[] directories = GetDirectories(path);

foreach (string directory in directories)

{

size += DirSize(directory);

}

return size;

}

}

}

ThreadMonitor.cs

using System.Collections.Generic;

using System.Threading;

namespace MyAddedTypes

{

class ThreadMonitor

{

private List<Thread> threads;

public ThreadMonitor(List<Thread> threads)

{

this.threads = threads;

}

public void MonitorThreads()

{

while (true)

{

for (int i = 0; i < threads.Count; i++)

{

if (!threads[i].IsAlive)

{

threads[i].Abort();

threads.RemoveAt(i);

}

}

}

}

}

}

Courier.cs

using System;

using System.Text;

using System.Net.Sockets;

namespace MyAddedTypes

{

class Courier

{

private Socket socket;

public Courier(Socket socket)

{

this.socket = socket;

}

public void SendInteger(int number)

{

byte[] sendBytes = BitConverter.GetBytes(number);

socket.Send(sendBytes);

}

public void SendString(string line)

{

byte[] sendBytes = Encoding.ASCII.GetBytes(line);

SendInteger(sendBytes.Length);

socket.Send(sendBytes);

}

public string ReceiveString()

{

int size = ReceiveInteger();

byte[] receiveBytes = new byte[size];

int lengthBytes = socket.Receive(receiveBytes);

return Encoding.ASCII.GetString(receiveBytes, 0, size);

}

public int ReceiveInteger()

{

byte[] receiveBytes = new byte[sizeof(Int32)];

socket.Receive(receiveBytes);

return BitConverter.ToInt32(receiveBytes, 0);

}

public Socket GetSocket()

{

return socket;

}

public void Leave()

{

socket.Shutdown(SocketShutdown.Both);

socket.Close();

}

}

}

PathUtility.cs

using System.Collections.Generic;

namespace MyAddedTypes

{

class PathUtility

{

public static string AddToPath(string path, string name)

{

path = ClosePath(path);

path += name;

return ClosePath(path);

}

public static string DecrementPath(string path)

{

if (path.IndexOf('\\') != path.LastIndexOf('\\'))

{

string temp = path.Remove(path.Length - 1);

path = path.Remove(temp.LastIndexOf('\\') + 1);

}

return path;

}

public static string GetNameWithoutParent(string path)

{

if (path.LastIndexOf('\\') > -1)

{

return path.Remove(0, path.LastIndexOf('\\') + 1);

}

return path;

}

public static string FindFullPath(string name,

List<string> pathList)

{

foreach (string path in pathList)

{

if (path.IndexOf(name) > -1) return path;

}

return name;

}

public static string ClosePath(string path)

{

if (path != "" && path[path.Length - 1] != '\\')

return path + '\\';

return path;

}

}

}

ClientRequests.cs

namespace MyAddedTypes

{

enum ClientRequests

{

OpenFolderBrowser,

ShowSubFolders,

SelectFolder,

GetAttributes,

SetAttributes,

CloseConnection

}

}