МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

автоматизованих систем обробки інформації та управління

**КУРСОВА РОБОТА**

з дисципліни

“ Об'єктно-орієнтоване програмування ”

на тему

"Синхронізація вказаних папок між двома комп’ютерами"

|  |  |  |
| --- | --- | --- |
| Керівник : доц. Муха І.П |  | Виконали :  Шелудько Д. М.,  Бугайов О.О.,  Ситник М.М. |
| Допущені до захисту |  | студент гр. ІП-61,ФІОТ |
| І\_\_\_І \_\_\_\_\_\_\_\_ 201\_ \_\_\_\_\_\_  підпис |  | 2 курс  № IП-6329  № ІП-6304  № ІП-6327 |
| Захистили з оцінкою  \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  оцінка підпис  І\_\_\_І \_\_\_\_\_\_\_\_\_ 201\_ |  | Підпис: |

Київ 2017

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ “КПІ”**

(назва вищого навчального закладу)

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Об’єктно-орієнтоване програмування

Напрям "Програмна інженерія"

Курс 2 Група ІП-63 Семестр 3

ЗАВДАННЯ

на курсову роботу студента

(прізвище, ім’я, по батькові)

1. Тема роботи

2. Строк здачі студентом закінченої роботи

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень )

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи |  |  |
| 2. | Підготовка ТЗ |  |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи |  |  |
| 4. | Розробка сценарію роботи програми |  |  |
| 6. | Узгодження сценарію роботи програми з керівником |  |  |
| 5. | Розробка (вибір) алгоритму рішення задачі |  |  |
| 6. | Узгодження алгоритму з керівником |  |  |
| 7. | Узгодження з керівником інтерфейсу користувача |  |  |
| 8. | Розробка програмного забезпечення |  |  |
| 9. | Налагодження розрахункової частини програми |  |  |
| 10. | Розробка та налагодження інтерфейсної частини програми |  |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу |  |  |
| 12. | Тестування програми |  |  |
| 13. | Підготовка пояснювальної записки |  |  |
| 14. | Здача курсової роботи на перевірку |  |  |
| 15. | Захист курсової роботи |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студенти

(підпис)

Керівник Муха І. П.

(підпис) (прізвище, ім’я, по батькові)

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

Анотація

Пояснювальна записка до курсової роботи: 70 сторінок, 16 рисунки, 20 таблиць, 4 посилання.

Об’єкт дослідження: синхронізація даних між комп’ютерами

Мета роботи: дослідження проблеми синхронізації даних між комп’ютерами. Написання програмного забезпечення спрямованого на вирішення проблеми синхронізації даних між комп’ютерами.

Вивчено методи мережевої взяємодії з метой обмінну даними між двома та більше користувачами. Приведені змістовні постановки задач, описано детальний процес розв’язання кожної з них.

Виконана програмна реалізація процесу синхронізації та обміну даними.

СИНХРОНІЗАЦІЯ ВКАЗАНИХ ПАПОК М­­­­ІЖ КОМПТ’ЮТЕРАМИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Кафедра автоматизованих систем обробки інформації та управління

|  |  |  |
| --- | --- | --- |
| **РОЗРОБИЛИ** |  | **ЗАТВЕРДИВ** |
| **Студенти**  \_\_\_\_\_\_\_\_\_\_  (підпис) (ініціали, прізвище)  “10” листопада 2017 р. |  | **Керівник**  \_\_\_\_\_\_\_\_\_\_ *І.П.Муха*  (підпис) (ініціали, прізвище)  “11” листопада 2017 р. |

Інтерпретатор мови програмування

**ТЕХНІЧНЕ ЗАВДАННЯ**

на виконання курсової роботи

з дисципліни:

«Об'єктно-орієнтоване програмування»

Шифр КПІ ІП-6109.ТЗ

на 14 сторінках

Київ 2017

ЗМІСТ

[1 найменування та область застосування 8](#_Toc501994711)

[2 підстави для розробки 9](#_Toc501994712)

[2.1 Перелік документів, на підставі яких ведеться розробка 9](#_Toc501994713)

[2.2 Найменування і умовне позначення теми розробки 9](#_Toc501994714)

[3 призначення розробки 10](#_Toc501994715)

[3.1 Функціональне призначення розробки 10](#_Toc501994716)

[3.2 Експлуатаційне призначення 10](#_Toc501994717)

[4 вимоги до програмного забезпечення 11](#_Toc501994718)

[4.1 Вимоги до функціональних характеристик 11](#_Toc501994719)

[4.1.1 Вимоги до складу виконуваних функцій 11](#_Toc501994720)

[4.1.2 Вимоги до організації вхідних даних 11](#_Toc501994721)

[4.1.3 Вимоги до організації вихідних даних 11](#_Toc501994722)

[4.2 Вимоги до надійності 11](#_Toc501994723)

[4.2.1 Вимоги до надійного функціонування програми 11](#_Toc501994724)

[4.2.2 Час відновлення після відмови 11](#_Toc501994725)

[4.2.3 Відмови через некоректні дії користувача 12](#_Toc501994726)

[4.3 Умови експлуатації 12](#_Toc501994727)

[4.3.1 Кліматичні умови експлуатації 12](#_Toc501994728)

[4.3.2 Вимоги до видів обслуговування 12](#_Toc501994729)

[4.3.3 Вимоги до кількості та кваліфікації персоналу 12](#_Toc501994730)

[4.4 Вимоги до складу і параметрів технічних засобів 12](#_Toc501994731)

[4.5 Вимоги до інформаційної і програмної сумісності 13](#_Toc501994732)

[4.6 Вимоги до маркування та упаковки 13](#_Toc501994733)

[4.7 Вимоги до транспортування та зберігання 13](#_Toc501994734)

[5 вимоги до програмної документації 14](#_Toc501994735)

[5.1 Попередній склад програмної документації 14](#_Toc501994736)

[6 техніко-економічні показники 15](#_Toc501994737)

[7 стадії та етапи розробки 16](#_Toc501994738)

[7.1 Стадії розробки 16](#_Toc501994739)

[7.2 Етапи розробки 16](#_Toc501994740)

[7.3 Зміст роботи по етапам 16](#_Toc501994741)

[8 порядок контролю і прийому 18](#_Toc501994742)

[8.1 Види досліджень 18](#_Toc501994743)

[8.2 Загальні вимоги до прийняття роботи 18](#_Toc501994744)

1. найменування та область застосування

Найменування – «Синхронізація зазначених папок між двома та більше пристроями».

Програма призначена до застосування користувачами операційних систем Microsoft Windows 7+, Linux, iOS 9.0+.

1. підстави для розробки
   1. Перелік документів, на підставі яких ведеться розробка

Підставою для проведення розробки являється наказ № 6 від 11 травня 2016 року. Наказ затверджено на засіданні кафедри АСОІУ, далі Замовник і прийнято до виконання студентоми Ситник М.М., Бугайов О.О., Шелудько Д.М., далі Виконавці, дата затвердження ТЗ.

* 1. Найменування і умовне позначення теми розробки

Найменування теми розробки – «Синхронізація зазначених папок між двома та більше пристроями».

Умовне позначення теми розробки (шифр теми) – «DML 1.0».

1. призначення розробки
   1. Функціональне призначення розробки

Функціональним призначенням розробки програми є надання користувачам можливості cинхронізування зазначених папок та файлів між двома пристроями.

* 1. Експлуатаційне призначення

Програма повинна експлуатуватися на об'єктах кафедри АСОІУ.

Кінцевими користувачами програми є власники компьютерів та телефонів, що бажають мати постійни й доступ до своїх файлів. Користувачами програми є власники комп’ютерів з встановленою операційною системою Microsoft Windows 7+, Linux, iOS 9.0+.

1. вимоги до програмного забезпечення
   1. Вимоги до функціональних характеристик
      1. Вимоги до складу виконуваних функцій

Програма повинна забезпечувати можливість виконання нижче наведених функцій:

а) Логін існуючого та реєстрація нового користувача;

б) Перегляд вмісту директорії;

в) Синхронізація вмісту директорії;

г) Обмін файлами;

* + 1. Вимоги до організації вхідних даних

На вхід подається логін та пароль користувача.

* + 1. Вимоги до організації вихідних даних

Вихідними даними для програми є інформація про структуру директорії, що синхронізується.

* 1. Вимоги до надійності
     1. Вимоги до надійного функціонування програми

Надійне функціонування програми повинно бути забезпечене виконанням замовником сукупності організаційно-технічних заходів, перелік яких наведено нижче:

а) організацією безперебійного струмопостачання технічних засобів;

б) використанням ліцензійного програмного забезпечення;

в) використанням своєчасно оновленого програмного забезпечення;

г) наявності стабільного Інтернет з’єднання.

* + 1. Час відновлення після відмови

Час відновлення після відмови, що спричинена збоєм електропостачання технічних засобів, не крахом операційної системи, не повинно перевищувати 1 години за умови виконання умов експлуатації технічних і програмних засобів.

Час відновлення після відмови, що спричинена неполадкою технічних засобів, крахом операційної системи, не повинно перевищувати часу, який потрібен на ліквідацію неполадок технічних засобів та переустановленням програмних засобів.

* + 1. Відмови через некоректні дії користувача

Відмови програми можливі внаслідок некоректних дій користувача системи. Задля запобігання відмов програми через вище вказані причини слід обмежити коло користувачів з правами доступу «Адміністратор» та на телефоні встановити пароль.

* 1. Умови експлуатації
     1. Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що заявлені до технічних засобів зокрема до умов їх експлуатації.

* + 1. Вимоги до видів обслуговування

Програма не потребує будь-яких видів обслуговування.

* + 1. Вимоги до кількості та кваліфікації персоналу

Мінімальна кількість персоналу, що необхідний для роботи програми, повинна складати не менше 1 штатної одиниці – користувач системи з правами «Адміністратор».

* 1. Вимоги до складу і параметрів технічних засобів

До комп’ютера, на якому має бути розгорнуто програмне забезпечення, висуваються наступні вимоги:

а) 64-розрядний процесор з тактовою частотою не нижче 2.5 ГГц;

б) об’єм оперативної пам’яті не менше 4 Гб;

в) об’єм жорсткого диску не менше 80 Гб;

г) Інтернет з’єднання зі швидкістю не меньше 512 кб/с.

* 1. Вимоги до інформаційної і програмної сумісності

Комп’ютер з операційною системою Microsoft Windows 7+, Linux та телефон з операційною системою iOS 9.0+.

* 1. Вимоги до маркування та упаковки

Вимоги до маркування та упаковки не висуваються.

* 1. Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

1. вимоги до програмної документації
   1. Попередній склад програмної документації

Склад програмної документації повинен включати в себе:

1. технічне завдання;
2. керівництво користувача.
3. техніко-економічні показники

Орієнтовна економічна ефективність не обчислюється.

1. стадії та етапи розробки
   1. Стадії розробки

Розробка повинна бути проведена в три стадії:

1. розробка технічного завдання;
2. робоче проектування;
3. впровадження.
   1. Етапи розробки

На стадії розробки технічного завдання повинен бути виконаний етап розробки, узгодження і затвердження технічного завдання.

На стадії робочого проектування повинен бути виконаний наведений нижче перелік робіт:

1. розробка програми;
2. тестування програми;
3. дослідження програми;
4. розробка програмної документації.

На стадії впровадження виконується передача програми кінцевому замовнику.

* 1. Зміст роботи по етапам

На етапі розробки технічного завдання повинні бути виконані наступні роботи:

1. постановка задачі;
2. визначення і уточнення вимог до технічних засобів;
3. визначення вимог до програми;
4. визначення стадій, етапів і строків розробки програми та документації на неї;
5. вибір мов програмування;
6. узгодження та затвердження технічного завдання.

На етапі розробки програми повинні бути виконані роботи по програмуванню і налагодженню програми.

На етапі тестування повинні бути виконані роботи по тестуванню усіх основних синтаксичних конструкцій мови програмування, що інтерпретується, в обох режимах роботи програми: режимі інтерпретації файлу та режимі REPL.

На етапі досліджень програми повинні бути виконані наведені нижче види робіт:

1. розробка, узгодження і затвердження програми і методик дослідження;
2. проведення досліджень;
3. коригування програми і програмної документації по результатам досліджень.

На етапі розробки програмної документації повинна бути виконана розробка програмних документів.

На етапі підготовки та передачі програми повинна бути виконана робота по підготовці та передачі програми і програмної документації в експлуатацію на об’єктах кафедри АСОІУ.

1. порядок контролю і прийому
   1. Види досліджень

Приймально-здавальні випробування програмного продукту мають проводитися згідно з розробленою виконавцем і узгодженою із замовником “Програмою та методикою випробувань” на об’єкті замовника.

* 1. Загальні вимоги до прийняття роботи

На основі проведених досліджень Виконавець спільно з Замовником підписують Акт прийому-здачі програми в експлуатацію.

**Пояснювальна записка  
до курсової роботи**

|  |  |
| --- | --- |
| на тему: |  |
|  |  |
|  | |

Київ 2017

зміст

[1 постановка задачі 5](#_Toc502256150)

[2 викладення використовуваних методів 6](#_Toc502256151)

[2.1 Основні поняття 6](#_Toc502256152)

[2.2 Розпаковка запиту 6](#_Toc502256153)

[2.3 Формування запиту 6](#_Toc502256154)

[2.4 Формування відповіді 6](#_Toc502256155)

[2.5 Розпаковка відповіді 7](#_Toc502256156)

[3 опис алгоритму 8](#_Toc502256157)

[3.1 Загальний алгоритм 8](#_Toc502256158)

[3.2 Алгоритм розпаковки запиту 9](#_Toc502256159)

[3.3 Алгоритм формулювання відповіді 9](#_Toc502256160)

[3.4 Алгоритм формування запиту 9](#_Toc502256161)

[3.5 Алгоритм розпаковки відповіді 10](#_Toc502256162)

[4 опис програмного забезпечення 11](#_Toc502256163)

[4.1 Опис діаграми класів програмного забезпечення клієнтської сторони 11](#_Toc502256164)

[4.2 Опис серверних класів 15](#_Toc502256165)

[5 тестування програмного забезпечення 20](#_Toc502256166)

[5.1 План тестування 20](#_Toc502256167)

[5.2 Приклади тестування 20](#_Toc502256168)

[5.2.1 Тестування при введенні некоректного логіна та пароля 20](#_Toc502256169)

[5.2.2 Тестування при проблемах з Інтернет з’єднанням 21](#_Toc502256170)

[5.2.3 Тестування при видалені файлів на клієнті 23](#_Toc502256171)

[5.2.4 Тестування при видалені файлів на сервері 24](#_Toc502256172)

[5.2.5 Тестування при відпраці неправильних запитів 25](#_Toc502256173)

[6 інструкція користувача 29](#_Toc502256174)

[6.1 Призначення програми 29](#_Toc502256175)

[6.2 Вимоги до системи 29](#_Toc502256176)

[6.3 Інструкція по користуванню програмою 29](#_Toc502256177)

[Додаток А Тексти програмного коду 33](#_Toc502256178)

вступ

Дана робота присвячена розробці програмного забезпечення синхронізації даних між двома та більше комп’ютерами.

Призначенням роботи є надання можливості синхронізації даних між комп’ютерами, що дає можливість отримати віддалений доступ до файлів та даних.

Програма працює за принципом клієнт-серверної взаємодії. На стороні сервера знаходиться база даних, в якій зберігається інформація щодо користувачів та їх даних. Клієнт використовується для з’єднання із сервером для подальшої взаємодії та отриманні чи відправленні даних із сервера та на сервер.

У випадку наявності стабільного Інтернет з’єднання, клієнтська сторона починає сесію взаємодії із сервером. По завершенню сесії, клієнт отримує останні оновлення пов’язані з його даними або відправляє свої дані на сервер.

Програмне забезпечення може використовуватися у всіх галузях людської діяльності .

# постановка задачі

Розробити програмне забезпечення для реалізації процесу синхронізації даних між двома та більше комп’ютерами.

У випадку, встановлення стабільного двостороннього Інтернет з’єднання починається клієнт-серверна взаємодія. На стороні сервера знаходиться база даних, в якій зберігається інформація щодо користувачів та їх даних. Клієнт використовується для з’єднання із сервером для подальшої взаємодії та отриманні чи відправленні даних із сервера та на сервер.

Вихідними даними для програми є структура директорії, що синхронізується.

# викладення використовуваних методів

## Основні поняття

Для взаємодії серверної сторони із базою даних, використовується Об'єктно-реляційне відображення (ORM). В даній роботі, ORM використовується для відображення клієнта в базі даних. Це дає змогу підвищити рівень абстракції взаємодії із базою даних.

Також для клієнт-серверної взаємодії був розроблений спеціальний протокол (DML). Він чітко визначає типи і структуру запитів та відповідей на них. DML є протоколом прикладного рівня і реалізується поверх TCP/IP.

## Розпаковка запиту

Розпаковка запиту – це процес перетворення запиту клієнта на масив, який складається з послідовності символів, що містяться між роздільником. Розпаковка запиту потрібна для подальшого аналізу та оброблення клієнтського запиту.

Вхідними даними для розпакувальника є послідовність байтів, що сервер отримав від клієнта. Під час розпакування, виділяються частини байтів, що містяться між роздільниками.

На виході розпакувальника, у випадку коли запит сформований правильно, повертається ступінчатий масив, що відображає структуру запиту.

## Формування запиту

Формування запиту – це процес перетворення окремих синтаксичних конструкцій на правильно сформульований та структурований масив байтів.

Вхідними даними для формувальника є тип запиту та параметри необхідні для правильного структурування запиту.

На виході формувальник, у випадку правильних вхідних даних повертає масив байтів, правильно сформованого і структурованого запиту.

## Формування відповіді

Формування відповіді – це процес компонування різних синтаксичних частин, що є відповіддю на оброблений запит.

Вхідними даними для формувальника є тип відповіді та параметри необхідні для правильної структури.

На виході формувальник, у випадку правильних вхідних даних повертає масив байтів, правильно сформованої і структурованої відповіді.

## Розпаковка відповіді

Розпаковка відповіді – це процес перетворення відповіді сервера на масив, який складається з послідовності символів, що містяться між роздільником. Розпаковка відповіді потрібна для подальшого аналізу та оброблення серверної відповіді.

Вхідними даними для розпакувальника є послідовність байтів, що клієнт отримав від сервера. Під час розпакування, виділяються частини байтів, що містяться між роздільниками.

На виході розпакувальника, у випадку коли відповідь сформована правильно, виконуються певні дії на стороні клієнта.

# опис алгоритму

## Загальний алгоритм

Розглянемо загальний алгоритм виконання інтерпретації коду.

1. ПОЧАТОК
2. Вибір режиму логіну або регістрації
3. Зчитування логіну та пароля
4. ЯКЩО логін та пароль введені невірно, ТО
   1. Видати повідомлення про помилку
5. ЯКЩО вибрано режим регістрації, ТО
   1. Відіслати запит про регістрацію на сервер
   2. Отримати відповідь від сервера
   3. ЯКЩО користувач вже існує, ТО
      1. Вивести повідомлення про помилку
   4. ІНАКШЕ
      1. Перейти до режиму логіну
6. ІНАКШЕ
   1. Відправити запит на сервер
   2. ЯКЩО користувача не знайдено, ТО
      1. Видати повідомлення про помилку
   3. ІНАКШЕ
      1. Відіслати запит на оновлення на сервер
7. Отримати відповідь від сервера
8. Перевірити наявність всіх файлів у клієнта
9. ЯКЩО файлу не існує АБО версія файлу застаріла, ТО
   1. Відправити запит про завантаження файлу
10. ІНАКШЕ
    1. Перевірити наявність всіх файлів у сервера
    2. ЯКЩО файлу не існує АБО версія файлу застаріла, ТО
       1. Завантажити файл на сервер
11. Вивести користувачу результат синхронізації
12. Очікувати повідомлення сервера або системи про зміну файлів
13. ЯКЩО отримано повідомлення про зміни файлів, ТО
    1. Перейти до п. 6.3.1
14. КІНЕЦЬ

## Алгоритм розпаковки запиту

1. ПОЧАТОК
2. Зчитування масиву байтів
3. Розділення байтів за роздільником
4. Формування ступінчатого масиву байтів
5. Перевірка запиту на правильність
6. ЯКЩО в форматі запиту є помилка, ТО
   1. Відправити повідомлення про помилку
7. ІНАКШЕ
   1. Виконати зазначені в запиті команди
   2. Сформувати відповідь
8. КІНЕЦЬ

## Алгоритм формулювання відповіді

1. ПОЧАТОК
2. Визначити тип відповіді та необхідні параметри
3. Сформувати масив байтів
4. Відправити клієнту масив
5. КІНЕЦЬ

## Алгоритм формування запиту

1. ПОЧАТОК
2. Визначити тип запиту та необхідні параметри
3. Сформувати масив байтів
4. Відправити запит серверу
5. КІНЕЦЬ

## Алгоритм розпаковки відповіді

1. ПОЧАТОК

2. Зчитування масиву байтів

3. Розділення байтів за роздільником

4. Формування ступінчатого масиву байтів

5. Перевірка відповіді на правильність

6. ЯКЩО в форматі відповіді є помилка, ТО

6.1. Відправити повідомлення про помилку

7. ІНАКШЕ

7.1.Виконати зазначені в запиті команди

8. КІНЕЦЬ

# опис програмного забезпечення

## Опис діаграми класів програмного забезпечення клієнтської сторони

Програмний додаток містить головний файл, 9 заголовних файлів та 9 файлів реалізації. У програмі міститься 9 класів.

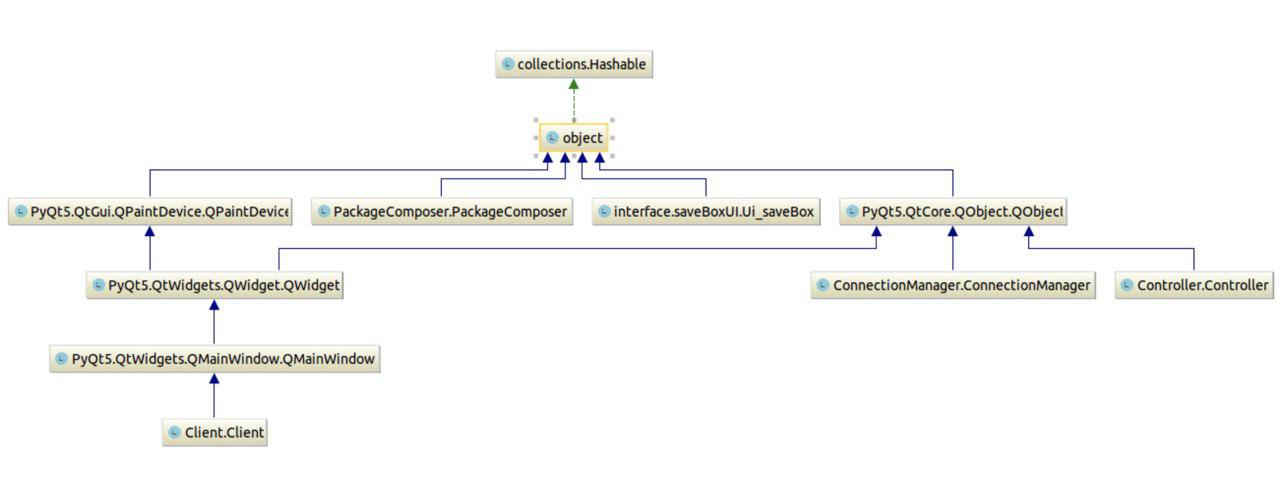


Рисунок 4.1 – UML-діаграма клієнта

Таблиця 4.1 – Опис клієнтських класів та методів

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовочний файл |
| 1 | Client | \_\_init\_\_ | Контструктор | - | - | Client.py |
| 2 | Client | startLoading | Полаток анімації завантаження | - | - | Client.py |
| 3 | Client | stopLoading | Кінець анімації завантаження | Результат останнього запису | - | Client.py |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | Client | startSync | Початок процесу синхронізації | - | - | Client.py |
| 5 | Client | waitForStartSync | Очікування синхронізації від сервера | - | - | Client.py |
| 6 | Client | showMessage | Вивід повідомлення про помилку | Повідомлення про помилку | - | Client.py |
| 7 | Client | CanselRegistration | Відміна регістрації | - | - | Client.py |
| 8 | Client | confirmRegistration | Початок регістрації | Логін,  Парорль | - | Client.py |
| 9 | Client | login | Логін користувача | Логін,  Пароль | - | Client.py |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | ConnectionManager | sendRequest | Відправка запиту | Тип запиту,  Аргументи запиту | - | ConnectionManager.py |
| 11 | ConnectionManager | getResponce | Отримання відповіді | - | - | ConnectionManager.py |
| 12 | Contoller | handle | Обробка выдповіді від сервера | Тип запиту,  Відповідь | - | Controller.py |
| 13 | PackageComposer | compose | Збірка данних у структурований запит | Тип запиту, Аргументи запиту | Структурований запит | PackageComposer.py |
| 12 | PackageComposer | disasemble | Розбиття відповіді на структуровану відповідь | Відповідь | Структурована відповідь | PackageComposer.py |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 13 | UI\_savebox | setup\_ui | Підготовка графічного інтерфейсу | Батько віджет | - | saveBoxUI |

Таблиця 4.2 – Опис стандартних клієнтських класів та методів

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | QWidget | \_\_init\_\_ | Підготовка віджета | Батько віджета | - | PyQt5.py |
| 2 | QObject | \_\_init\_\_ | Підготовка  Об’єкту | - | - | PyQt5.py |
| 3 | - | timeFromDatetime | Створення марки дати | Datetime | Timestamp | time.py |
| 4 | - | split | Розбиття об’єекту на частини | Iterable | Масив Iterable | PythonDefaultLibrary |
| 5 | - | print | Вивід у консоль | Дані для виводу | - | PythonDefaultLibrary |
| 6 | Datetime | getDelta | Отримати результат різниці datetime | - | timeStamp | datetime.py |
| 7 | Socket | socket | Отримання об’єкту сокет | Тип сонета, Протокол | socket | socket.py |

## Опис серверних класів

Опис серверних класів програмного забезпечення та їх методів наведено у таблиці 4.3.



Таблиця 4.3 – Опис серверних класів та методів

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | EServer | Initialize | Ініціалізація серверу ІП адресою та портом | Відсутні | - | EServer.cs |

Продовження таблиці 4.3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | EServer | Start | Сервер починає свою роботу. Починає слухати певний порт | Відсутні | - | EServer.cs |
| 3 | EServer | Initialize | Ініціалізація серверу ІП адресою та портом | Відсутні | - | EServer.cs |
| 4 | PackageAssembly | Assemble | Збір відповіді клієнту | Відсутні | - | PackageAssembly.cs |
| 5 | RequestProcessor | SendFile | Створення файлу | Відсутні | - | RequestProcessor.cs |

Продовження таблиці 4.3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 6 | RequestProcessor | SendDirectory | Створення директорії на сервері | Відсутні | - | RequestProcessor.cs |
| 7 | RequestProcessor | IsAlive | Логгінг | Відсутні | - | RequestProcessor.cs |
| 8 | RequestProcessor | Register | Регістрація нового користувача | Відсутні | - | RequestProcessor.cs |

Продовження таблиці 4.3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 9 | RequestProcessor | SendFile | Відсилання файлу клієнту | Відсутні | - | RequestProcessor.cs |
| 10 | ResponceProcessor | GetFile | Відповідь на запит | Відсутні | Повертає відповіді клієнту | ResponceProcessor.cs |
| 11 | ResponceProcessor | GetLastUpdate | Повертає останок онавлення | Відсутні | Повертає відповіді клієнту | ResponceProcessor.cs |
| 12 | ResponceProcessor | SendDirectory | Відповідь | Відсутні | Повертає відповіді клієнту | ResponceProcessor.cs |

Таблиця 4.4 – Опис стандартних серверних класів та методів

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 |  | GetLine | Зчитування інформації із потоку | istream &is – поток зчитування. String &str – рядок, у який відбувається зчитування | Функція повертає значення типу istream & - поток зчитування | string |
| 2 | vector | ClearScrean | Очищення масиву | Відсутні | Функція не повертає результату | vector |
| 3 | vector | Add | Вставка елементу в кінець масиву | T element – елемент для вставки (тип співпадає з типом вектора) | Функція не повертає результату | vector |
| 4 | vector | IsEmpty | Перевірка, чи є масив порожнім | Відсутні | Функція повертає результат типу bool. True, якщо масив порожній, False – якщо ні | vector |

# тестування програмного забезпечення

## План тестування

Для перевірки працездатності програмного забезпечення було проведено його тестування. Програмне забезпечення повинне правильно обробляти вхідні дані, в разі їх некоректності сповіщати про це користувача. Програма повинна адекватно реагувати на введення некоректного шляху до файлу з кодом. В разі введення коду, який містить лексичну помилку, програма повинна сповіщати про це користувача. В разі введення коду, який містить синтаксичну помилку, програма повинна сповіщати про це користувача. Також програма повинна коректно обробляти усі нештатні ситуації, які можуть виникнути в ході обрахунків, наприклад, ділення на нуль, знаходження визначника неквадратної матриці тощо.

План тестування програмного забезпечення на відповідність усім вищесказаним вимогах наведено нижче.

1. Тестування при введенні некоректного логіну та пароля
   1. Тестування при веденні заборонених символів
   2. Тестування при веденні неіснуючого логіну та пароля
2. Тестування при проблемах з Інтернет з’єднанням
   1. Тестування при повільному Інтернет з’єднанні
   2. Тестування при втраті Інтернет з’єднання
3. Тестування при формуванні неправильних запитів
4. Тестування при видалені файлів під час роботи програми
   1. Тестування при видалені файлів на сервері
   2. Тестування при видалені файлів на клієнтській стороні

## Приклади тестування

Тестування програмного забезпечення було проведено згідно з планом, наведеним у п. 5.1. В ході тестування було отримано наступні результати.

### Тестування при введенні некоректного логіна та пароля

Після введення заборонених було отримано повідомлення, яке зображено на рисунку 5.1.

Результати тестування при введенні некоректного логіна та пароля наведено у таблиці 5.1.

Таблиця 5.1 – Приклад роботи програми при введені некоректного логіна та пароля

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення неіснуючого логіну та пароля |
| Початковий стан програми | Запущений клієнт і сервер |
| Вхідні дані | Логін-0000 Пароль-0000 |
| Схема проведення тесту | Введення неіснуючого логіна та пароля у відповідні поля програми |
| Очікуваний результат | Повідомлення про помилку  зчитування з файлу |
| Стан програми після проведення випробувань | Видано помилку «UserNotExists» |

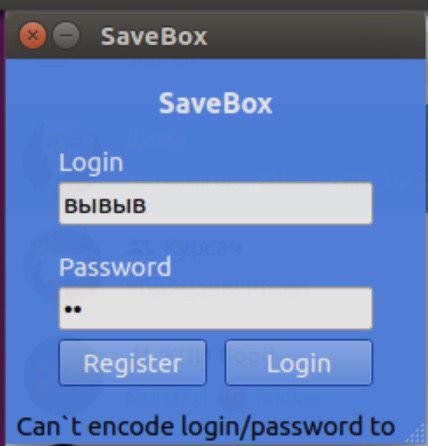


Рисунок 5.1 - Тестування при введенні заборонених символів у поля логіна та пароля

### Тестування при проблемах з Інтернет з’єднанням

Після проведення тесту при раптовій втраті Інтернет з’єднання було отримано повідомлення, зображене на рисунку 5.2.

Результати тестування при наявності проблем з Інтернет з’єднанням наведені в таблиці 5.2.

Таблиця 5.2 – Приклад роботи програми при проблемах з Інтернет з’єднанням

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість роботи програми при проблемах з Інтернет з’єднанням |
| Початковий стан програми | Запущений клієнт і сервер |
| Вхідні дані | - |
| Схема проведення тесту | Тестування при повільному Інтернет з’єднанні.  Тестування при раптовому розриві Інтернет з’єднанні. |
| Очікуваний результат | Корректна робота програми  Повідомлення про розрив з’єднання |
| Стан програми після проведення випробувань | Програма продовжила свою роботу  Програма вивела повідомлення про помилку |

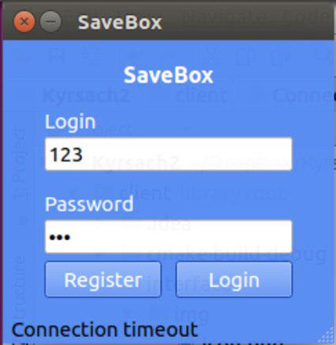


Рисунок 5.2 - Тестування при раптовому Інтернет роз’єднанні

### Тестування при видалені файлів на клієнті

Таблиця 5.3 – Приклад роботи програми при введені некоректного логіна та пароля

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість роботи програми при проблемах з Інтернет з’єднанням |
| Початковий стан програми | Запущений клієнт і сервер |
| Вхідні дані | - |
| Схема проведення тесту | Тестування при видалені файлу на клієнті |
| Очікуваний результат | Корректна робота програми  Завантаження файлу на клієнт |
| Стан програми після проведення випробувань | Програма виконала завантаження |

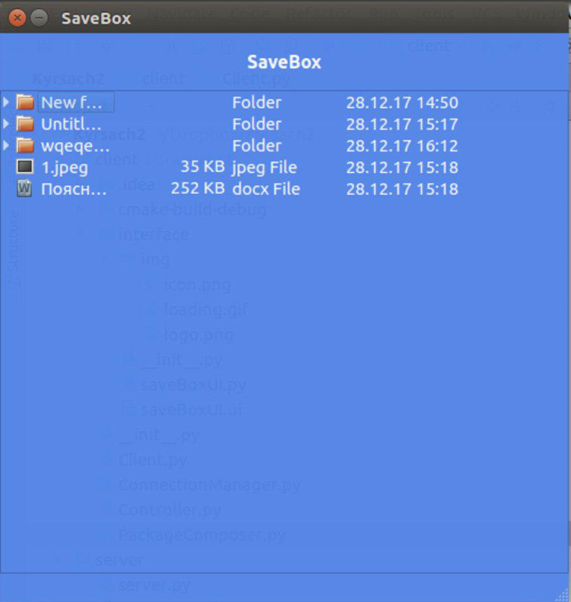


Рисунок 5.3 - Тестування при видалені файлів на клієнті

### Тестування при видалені файлів на сервері

Таблиця 5.4 – Приклад роботи програми при введені некоректного логіна та пароля

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість роботи програми при проблемах з Інтернет з’єднанням |
| Початковий стан програми | Запущений клієнт і сервер |
| Вхідні дані | - |
| Схема проведення тесту | Тестування при видалені файлу на сервері |
| Очікуваний результат | Корректна робота програми  Завантаження файлу на сервер |
| Стан програми після проведення випробувань | Програма виконала завантаження |

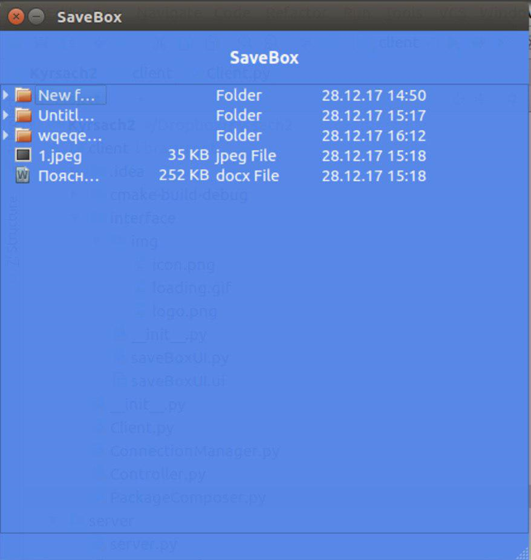
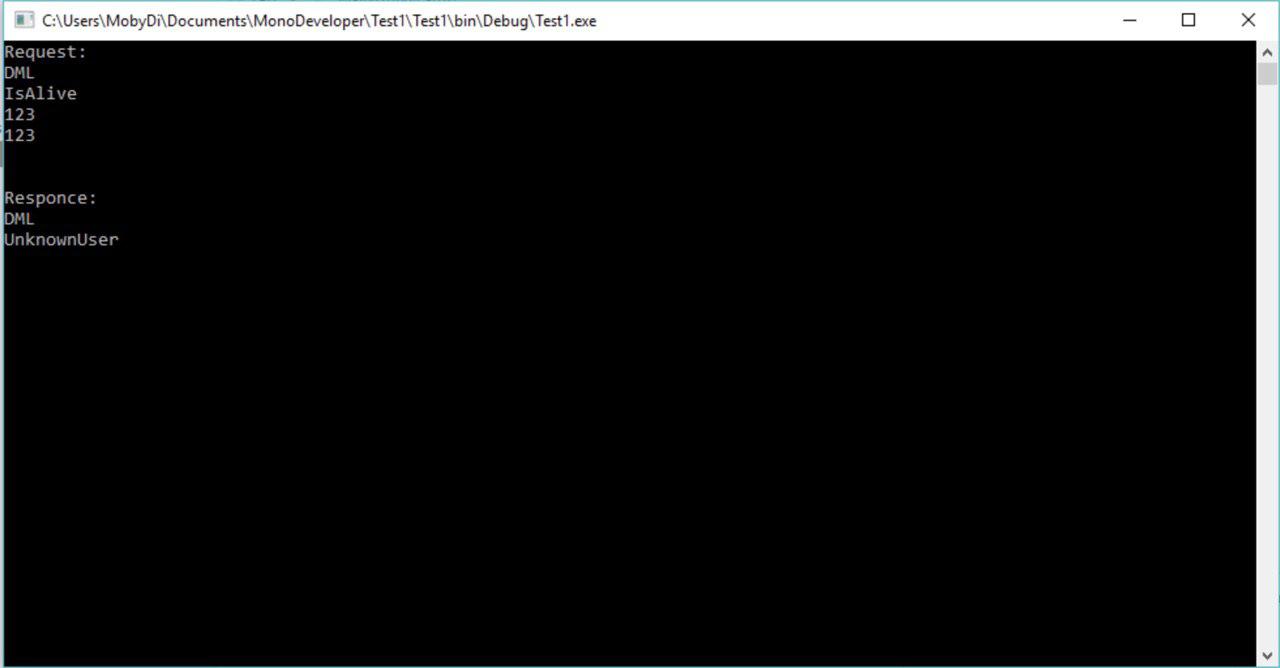


Рисунок 5.2 - Тестування при видалені файлу на сервері

### Тестування при відпраці неправильних запитів

Таблиця 5.5 – Приклад роботи програми при введені некоректного логіна та пароля

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість роботи програми при проблемах з Інтернет з’єднанням |
| Початковий стан програми | Запущений клієнт і сервер |
| Вхідні дані | Неправильний логгінг клієнта |
| Схема проведення тесту | Тестування при неправильному логгінгу клієнта |
| Очікуваний результат | Корректна робота програми  Відправка відповіді |
| Стан програми після проведення випробувань | Програма виконала відправку відповіді |

 Рисунок 5.3 - Тестування при неправильнму логінгу

Таблиця 5.6 – Приклад роботи програми при відправці некоректного пакету з клієнтської сторони

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість роботи програми при проблемах з Інтернет з’єднанням |
| Початковий стан програми | Запущений клієнт і сервер |
| Вхідні дані | Неправильний логгінг клієнта |
| Схема проведення тесту | Тестування при неправильному форматі пакету |
| Очікуваний результат | Корректна робота програми  Відправка відповіді |
| Стан програми після проведення випробувань | Програма виконала відправку відповіді |



Рисунок 5.4 - Тестування при неправильнму форматі пакету

Таблиця 5.7 – Приклад роботи програми при неіснуючому файлі

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість роботи програми при проблемах з Інтернет з’єднанням |
| Початковий стан програми | Запущений клієнт і сервер |
| Вхідні дані | Неправильний доступ до файлу |
| Схема проведення тесту | Тестування при неправильному логгінгу клієнта |
| Очікуваний результат | Корректна робота програми  Відправка відповіді |
| Стан програми після проведення випробувань | Програма виконала відправку відповіді |

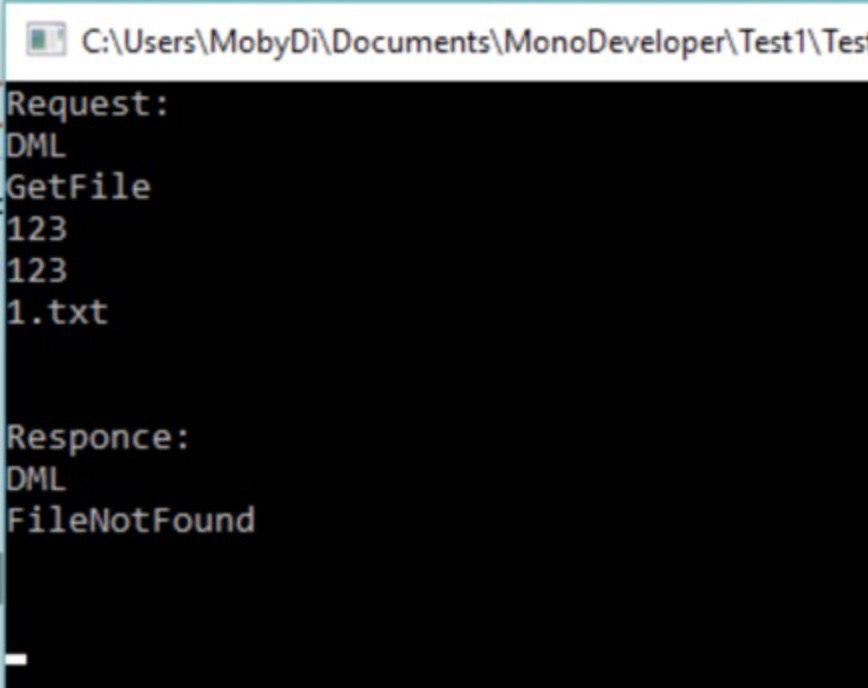


Рисунок 5.5 - Тестування при неправильнму логінгу

# інструкція користувача

## Призначення програми

Головним призначенням програми є надання можливості синхронізації файлів різних користувачів на 2 і більше комп’ютерах. Програма може використовуватися у всіх галузях людської діяльності, задля забезпечення комфортності передачі даних.

## Вимоги до системи

* Операційна система: Windows 7 або новіша
* RAM: 2 Gb або більше
* Вільне місце на диску: 5 Gb або більше
* Процесор: Intel Pentium 1.6 GHz або кращий
* Дисплей: 1024 x 768 або більший

## Інструкція по користуванню програмою

Програма складається з клієнтської та серверної сторони.

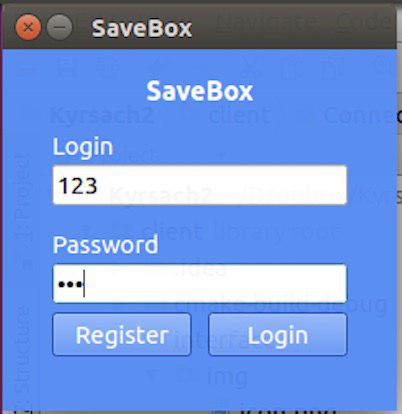
Для того щоб запустити програму, необхідно ввести логін та пароль існуючого користувача, або ж зареєструвати нового користувача. Приклад використання програми на даному етапі показано на рисунку 6.1.

Рисунок 6.1 – Авторизація користувача

Після цього програма починає автоматично слідкувати за папкою “SaveBox”.

З цього моменту починається процес синхронізації користувацьких файлів. Дані про зміну файлів відправляються на сервер, там вони обробляються і зберігаються. В такому випадку усі клієнти, що підключені за даним логіном та паролем починають порівнювати свої файли, та файли на сервері. Вікно із заупщеним процесом синхронізації зобрежено на рисунку 6.3.

Рисунок 6.3 – Вікно із запущеним процесом синхронізації

висновки

На етапі підготовки до написання курсової роботи були детально розглянуті та вивчені усі аспекти написання програми синхронізації файлів користувачів, які знаходяться на 2 і більше комп’ютерах. Обрано структуру запитів і відповідей, для роботи програми за принципом клієнт-сервер. Також були обрані мові програмування для написання усіх складових проекту.

Наступним етапом було проектування програми, під час чого було створено ієрархію класів за допомогою яких відбувалась взаємодія клієнта та сервера, а також ієрархію класів, за допомогою яких проводилось спостереження за файловою системою користувача і сервера.

Після цього було проведено програмну реалізацію вивчених аспектів, перевірку їх працездатності при різних вхідних даних. Після завершення написання програми було проведено її тестування та перевірка правильності роботи. Було встановлено, що програма коректно обробляє будь-які вхідні дані та виконує поставлену задачу.

У ході виконання курсової роботи було ретельно вивчено основні аспекти написання програм клієнт-сверверної архітектури, а також програм, які виконують синхронізацію файлів. Було розроблено програмне забезпечення, яке дає змогу синхронізувати файли на різних комп’ютерах за допомогою мережі Інтернет.

перелік посилань

1. Молчанов А. Ю. Системное программное обеспечение: Учебник для вузов. 3-е изд. - СПб.: Питер, 2010.
2. Гордеев А. В., Молчанов А. Ю. Системное программное обеспечение. — СПб.: Питер, 2001. — 734 с.

Додаток А Тексти програмного коду

*студентів групи ІП-63 ІІ курсу*

*Шелудько Д.М.*

*Бугайова О.О.*

*Ситника М.М.*

(Обсяг програми (документа), арк., Кб)

*112 арк, 108 Кб*

(Вид носія даних)

*CD-RW*

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення інтерпретатора мови програмування*

**Додаток Б  
до курсової роботи**

|  |  |
| --- | --- |
| на тему: | *Граматика розробленої мови програмування* |
|  |  |
|  | |

Київ 2017

Client.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Server.ClientModel

{

class Client

{

public int Id { get; set; }

//To store a client working directory

public string WorkingDirectory { get; private set; }

//To save a client last update

public DateTime LastUpdate { get; set; }

public string Login { get; private set; }

public string Password { get; private set; }

//Add a non-required parametr

public Client(string login,string password,string workingDirectory = default(string))

{

Login = login;

Password = password;

WorkingDirectory = CloudConfigs.WorkingDirectoryConfig.GetWorkingDirectory() + login;

LastUpdate = DateTime.Now;

}

public Client()

{

Login = default(string);

Password = default(string);

WorkingDirectory = default(string);

LastUpdate = DateTime.Now;

}

}

}

ClientConfiguration

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity.ModelConfiguration;

namespace Server.Configurations

{

class ClientConfigurations:EntityTypeConfiguration<ClientModel.Client>

{

public ClientConfigurations()

{

Property(d => d.Login).IsRequired().HasMaxLength(20);

Property(d => d.Password).IsRequired();

Property(d => d.LastUpdate).IsRequired();

}

}

}

ClientContext

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity;

namespace Server

{

class ClientContext:DbContext

{

public DbSet<ClientModel.Client> Clients { get; set; }

/\* public IQueryable<ClientModel.Client> GetClients()

{

using (var context = new ClientContext())

{

return context.Set<ClientModel.Client>();

}

}\*/

}

}

PackageAssembly

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Server.PackageComposer

{

class PackageAssembly

{

//enum of all supported commands

string assembly = "DML\r\n";

Enums.DMLResponce type;

public PackageAssembly(Enums.DMLResponce responce)

{

type = responce;

switch(responce)

{

case Enums.DMLResponce.BadRequest:

{

assembly += "BadRequest\r\n\r\n";

break;

}

case Enums.DMLResponce.FileNotFound:

{

assembly += "FileNotFound\r\n\r\n";

break;

}

case Enums.DMLResponce.Ok:

{

assembly += "Ok\r\n";

break;

}

case Enums.DMLResponce.RegistrationOk:

{

assembly += "RegistrationOk\r\n\r\n";

break;

}

case Enums.DMLResponce.UnknownPackage:

{

assembly += "UnknownPackage\r\n\r\n";

break;

}

case Enums.DMLResponce.UnknownUser:

{

assembly += "UnknownUser\r\n\r\n";

break;

}

case Enums.DMLResponce.UserExists:

{

assembly += "UserExists\r\n\r\n";

break;

}

case Enums.DMLResponce.UserIsAlreadyExists:

{

assembly += "UserIsAlreadyExists\r\n\r\n";

break;

}

case Enums.DMLResponce.UserNotFound:

{

assembly += "UserNotFound\r\n\r\n";

break;

}

case Enums.DMLResponce.LastUpdates:

{

assembly += "LastUpdates\r\n";

break;

}

case Enums.DMLResponce.Sucess:

{

assembly += "Sucess\r\n\r\n";

break;

}

default:

{

assembly += "BadRequest\r\n\r\n";

break;

}

}

}

public byte[] Assemble()

{

return Encoding.UTF8.GetBytes(this.assembly);

}

public override string ToString()

{

return assembly;

}

public byte[] AddFile(byte[] file)

{

//DML\r\nOk\r\n

byte[] first = Encoding.UTF8.GetBytes(assembly);

//\r\n\r\n

byte[] end = Encoding.UTF8.GetBytes("\r\n\r\n");

//our responce

byte[] result = new byte[file.Length + first.Length + end.Length];

Buffer.BlockCopy(first, 0, result, 0, first.Length);

Buffer.BlockCopy(file, 0 , result, first.Length,file.Length);

Buffer.BlockCopy(end, 0, result, first.Length+file.Length, end.Length);

/\*//add the first part to the result

for(int i=0;i<first.Length;i++)

{

result[i] = first[i];

}

//offset

int ptr = first.Length;

//add file part using offset

for (int i = 0;i<file.Length;i++)

{

result[i + ptr] = file[i];

}

//inc the offset

ptr += file.Length;

//add \r\n\r\n

for(int i=0;i<end.Length;i++)

{

result[i + ptr] = end[i];

}

\*/

assembly = Encoding.UTF8.GetString(result);

return result;

}

public void AddUpdates(DateTime dateTime, string path, string clientDirectory)

{

assembly += Convert.ToString(dateTime);

//assembly += "\r\n";

string[][] allUpdates = CloudConfigs.WorkingDirectoryConfig.GetLastUpdate(path, clientDirectory);

for (int i = 0; i < allUpdates.Length; i++)

{

for (int j = 0; j < allUpdates[i].Length; j++)

{

assembly += "\r\n" + allUpdates[i][j];

}

}

assembly += "\r\n\r\n";

assembly.Replace('\\','/');

}

}

}

PackageDissasembly

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Server.PackageComposer

{

class PakageDisassembly

{

//DML package commands

public enum DMLCommands { Autorization = 0, GetLastUpdate, GetFile, SendFile }

//Package

private byte[] package;

private byte[][] unpack;

public PakageDisassembly(byte[] package)

{

if (package != default(byte[]) && package.Length > 3)

{

//need to check whether it is a DML package

byte[] dml = new byte[3];

Buffer.BlockCopy(package, 0, dml, 0, 3);

string DML = Encoding.UTF8.GetString(dml);

//Check if we have a DML protocol if not throw the exception

if (DML != "DML")

{

UnknownPakageException exception = new UnknownPakageException();

throw exception;

}

else

{

this.package = package;

}

}

else

{

UnknownPakageException exception = new UnknownPakageException();

throw exception;

}

//To check if it is a \r in the end of string[]

bool flags = true;

byte[][] pack = SpliteRN(package);

//Console.WriteLine();

//Console.WriteLine("Split the package:");

/\* for(int i=0;i<pack.Length;i++)

{

//Console.WriteLine("{0:x}",Encoding.UTF8.GetString(pack[i]));

}\*/

//Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//If package structure is inccorect

if (!flags && pack == null)

{

UnknownPakageException ex = new UnknownPakageException();

throw ex;

}

else

{

//Inittialize a package array field

unpack = pack;

}

}

private byte[][] SpliteRN(byte[] array)

{

try

{

string s = Encoding.UTF8.GetString(array);

//the result of caling the function

byte[][] result = default(byte[][]);

//Separator

byte[] separator = Encoding.UTF8.GetBytes("\r\n");

List<int> positions = new List<int>();

int count = 0;

for (int i = 0; i < array.Length; i++)

{

if (count == 5)

{

//check if it is a directory

if (i != array.Length - 4)

{

positions.Add(array.Length - 1);// - \r\n\r\n - 1

}

break;

}

if (array[i] == separator[0])

{

if (i < array.Length - 1 && array[i + 1] == separator[1])

{

positions.Add(i);

count++;

if(count == 4)

{

if(i == array.Length - 4)

{

break;

}

}

}

}

}

result = new byte[positions.Count][];

int currPosition = 0;

int offset = 0;

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

{

//current position

currPosition = positions[i];

//initialize each array

result[i] = new byte[currPosition - offset];

for(int j = 0;j<result[i].Length;j++)

{

result[i][j] = array[j + offset];

}

//add \r\n

offset = currPosition + 2;

}

return result;

}

catch

{

return default(byte[][]);

}

}

//To get an unpack package array

public byte[][] Unpack()

{

return unpack;

}

}

}

ApiManager.swift

//

// APIManager.swift

// DML

//

// Created by Equinox on 27.12.2017.

// Copyright © 2017 Equinox. All rights reserved.

//

import Foundation

import Starscream

class APIManager {

enum RequestType {

case logIn

case register

}

struct RequestResolver {

let fulfill: (Any?)->()

let reject: (Error)->()

}

typealias Request = (id: UInt32, type: RequestType, resolver: RequestResolver)

// MARK: - Singletone bullshit

static let sharedInstance = APIManager()

private init() { }

// MARK: - Properties

lazy var socket: WebSocket = {

let socket = WebSocket(url: URL(string: "ws://212.22.192.242:8050/")!)

socket.delegate = self

return socket

}()

var requestsBufferArray: [APIManager.Request] = []

func register(withLogin login: String, password: String, withCompetion completion: @escaping (Error?)->()) {

let id = arc4random\_uniform(1000)

let registerRequest: Request = (id: id, type: .register, resolver: RequestResolver(fulfill: { (responseData) in

completion(nil)

}, reject: { (error) in

completion(error)

}))

requestsBufferArray.append(registerRequest)

socket.write(string: "\(id)\r\nDML\r\nRegister\r\n\(login)\r\n\(password)\r\n\r\n")

completion(nil)

}

func logIn(withLogin login: String, password: String, withCompetion completion: @escaping (Error?)->()) {

let id = arc4random\_uniform(1000)

let logInRequest: Request = (id: id, type: .logIn, resolver: RequestResolver(fulfill: { (responseData) in

completion(nil)

}, reject: { (error) in

completion(error)

}))

requestsBufferArray.append(logInRequest)

socket.write(string: "DML\r\nIsAlive\r\n\(login)\r\n\(password)\r\n\r\n")

completion(nil)

}

}

extension APIManager: WebSocketDelegate {

func websocketDidConnect(socket: WebSocketClient) {

print("Socket connected")

}

func websocketDidDisconnect(socket: WebSocketClient, error: Error?) {

socket.connect()

}

func websocketDidReceiveMessage(socket: WebSocketClient, text: String) {

print("Received message:", text)

}

func websocketDidReceiveData(socket: WebSocketClient, data: Data) {

let exampleSecond = data as Any

//let datasecond = String((exampleSecond as AnyObject).characters.filter { !"DML\r\n \r\n\r\n".characters.contains($0) })

if let string = String(data: data, encoding: .utf8) {

let dataListening = String(string.replacingOccurrences(of: "DML\r\n \r\n\r\n", with: ""))

let repoList: [String] = [dataListening]

print("Received data: \(repoList)")

} else {

print("not a valid UTF-8 sequence")

}

}

// get id

// get request

// get type

// switch type

// run resolver with errr or valid data

}

Constants.swift

//

// Constants.swift

// DML

//

// Created by Equinox on 28.12.2017.

// Copyright © 2017 Equinox. All rights reserved.

//

import UIKit

class Constants: NSObject {

let filmNames = ["Geostorm", "IT", "Blade Runner 2049"]

let genre: [String] = ["action.jpg, triller.jpg, fantasy.jpg", "horror.jpg", "action.jpg, triller.jpg, fantasy.jpg"]

let country = "United States"

let director = ["Dean Davline", "Andy Muchietti", "Denis Villeneuve"]

let duration = [109, 108, 144]

let language = ["English", "English", "English"]

}

AppDelegate.swift

//

// AppDelegate.swift

// DML

//

// Created by Equinox on 12.12.2017.

// Copyright © 2017 Equinox. All rights reserved.

//

import UIKit

let userLogin = "misha"

let userPassword = "123"

@UIApplicationMain

class AppDelegate: UIResponder, UIApplicationDelegate {

var window: UIWindow?

func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

// Override point for customization after application launch.

return true

}

func applicationWillResignActive(\_ application: UIApplication) {

// Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.

// Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this method to pause the game.

}

func applicationDidEnterBackground(\_ application: UIApplication) {

// Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.

// If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.

}

func applicationWillEnterForeground(\_ application: UIApplication) {

// Called as part of the transition from the background to the active state; here you can undo many of the changes made on entering the background.

}

func applicationDidBecomeActive(\_ application: UIApplication) {

// Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.

}

func applicationWillTerminate(\_ application: UIApplication) {

// Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.

}

}

ConnectionManager.py

import socket

import threading

import queue

from PyQt5 import QtCore

from PackageComposer import PackageComposer

from Controller import Controller

class ConnectionManager(QtCore.QObject):

startLoading = QtCore.pyqtSignal()

stopLoading = QtCore.pyqtSignal(str, str)

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.composer = PackageComposer()

self.controller = Controller()

self.requestQueue = queue.Queue()

self.requestManagerThread = threading.Thread(target=self.requestManager, daemon=True)

self.requestManagerThread.start()

self.requestLock = threading.Lock()

def requestManager(self):

while True:

if threading.active\_count() == 3:

request = self.requestQueue.get()

request.start()

def sendRequest(self, showLoading=False, requestType='IsAlive', \*\*kwargs):

requestThread = threading.Thread(

target=self.\_\_sendRequest\_\_,

args=(showLoading, requestType),

kwargs=kwargs, daemon=True

)

self.requestQueue.put(requestThread)

def \_\_sendRequest\_\_(self, showLoading=False, requestType='IsAlive', \*\*kwargs):

if showLoading:

self.startLoading.emit()

self.requestLock.acquire()

try:

clientSock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

clientSock.settimeout(5)

clientSock.connect(('192.168.1.147', 8045))

clientSock.settimeout(None)

except Exception as e:

print('Error in connecting[', e.args, ']')

if showLoading:

self.stopLoading.emit(requestType, '')

self.requestLock.release()

return

try:

clientSock.sendall(self.composer.compose(requestType, \*\*kwargs))

print(requestType, kwargs)

response = self.\_\_getResponse\_\_(clientSock)

except Exception as e:

print('Error in connecting[', e.args, ']')

if showLoading:

self.stopLoading.emit(requestType, '')

self.requestLock.release()

return

response = self.composer.disassmeble(requestType, response)

self.controller.handle(requestType, response, \*\*kwargs)

clientSock.close()

if showLoading:

self.stopLoading.emit(requestType, response[1])

self.requestLock.release()

def \_\_getResponse\_\_(self, clientSock):

response = b''

size = int.from\_bytes(clientSock.recv(4), byteorder='little')

while True:

response += clientSock.recv(1024)

if len(response) == size and response.endswith('\r\n\r\n'.encode('UTF-8')):

break

print(response)

return response[:-4]

Controlers.py

from PyQt5 import QtCore

import os

import datetime

import time

class Controller(QtCore.QObject):

startSync = QtCore.pyqtSignal()

getFile = QtCore.pyqtSignal(str, object)

sendFile = QtCore.pyqtSignal(str)

sendDirectory = QtCore.pyqtSignal(str)

showMessage = QtCore.pyqtSignal(str)

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.saveFolder = os.path.expanduser('~') + '/SaveBox/'

def handle(self, requestType, response, \*\*kwargs):

if response[1] == 'BadRequest':

self.showMessage.emit('BadRequest')

return

if requestType == 'IsAlive':

if response[1] == 'UserExists':

self.startSync.emit()

#self.showMessage.emit(requestType+' '+response[1])

return

if requestType == 'GetLastUpdate':

for path in response[3:]:

serverCreationTime = path.split('\n')[1]

serverCreationTime = datetime.datetime.strptime(serverCreationTime, '%m/%d/%Y %H:%M:%S')

path = path.split('\n')[0]

path = path.replace('\\', '/')

absolutePath = os.path.join(self.saveFolder, path)

if path[-1] == '/':

os.makedirs(absolutePath, exist\_ok=True)

elif not os.path.exists(absolutePath):

self.getFile.emit(path, serverCreationTime)

else:

clientCreationTime = os.path.getmtime(absolutePath)

clientCreationTime = datetime.datetime.fromtimestamp(clientCreationTime)

delta = serverCreationTime - clientCreationTime

if delta.total\_seconds() > 0:

self.getFile.emit(path, serverCreationTime)

for root, dirs, files in os.walk(self.saveFolder):

relativeRootPath = os.path.relpath(root, self.saveFolder)

isExist = False

for path in response[3:]:

path = path.split('\n')[0]

if relativeRootPath == path[:-1]:

isExist = True

break

if not isExist and relativeRootPath != '.':

self.sendDirectory.emit(relativeRootPath)

for file in files:

isExist = False

absoluteFilePath = os.path.join(root, file)

relativeFilePath = os.path.relpath(absoluteFilePath, self.saveFolder)

for path in response[3:]:

serverCreationTime = path.split('\n')[1]

path = path.split('\n')[0]

if path == relativeFilePath:

clientCreationTime = os.path.getmtime(absoluteFilePath)

clientCreationTime = datetime.datetime.fromtimestamp(clientCreationTime)

serverCreationTime = datetime.datetime.strptime(serverCreationTime, '%m/%d/%Y %H:%M:%S')

delta = serverCreationTime - clientCreationTime

if delta.total\_seconds() >= 0:

isExist = True

if not isExist:

self.sendFile.emit(relativeFilePath)

#self.showMessage.emit(requestType+' '+response[2])

return

if requestType == 'GetFile':

dirs = kwargs['filename'].rsplit('/', 1)

if len(dirs) == 2:

os.makedirs(os.path.join(self.saveFolder,dirs[0]), exist\_ok=True)

file = open(os.path.join(self.saveFolder, kwargs['filename']), 'wb')

file.write(response[2])

serverCreationTime = kwargs['serverCreationTime']

timestamp = time.mktime(serverCreationTime.timetuple())

os.utime(os.path.join(self.saveFolder, kwargs['filename']), (timestamp, timestamp))

#self.showMessage.emit(requestType+' '+kwargs['filename']+' '+response[1])

return

#self.showMessage.emit(requestType+' '+';'.join(response))