



**Софийски университет „Св. Кл. Охридски“**

Факултет по математика и информатика

*Бакалавърска програма  
„Софтуерно инженерство“*



**Курс: Обектно-ориентирано програмиране с C#.NET**

*Зимен семестър, 2016/2017 год.*

**Документация на системата**

**BankCardTokenization**

*Автор: Драгомир Евгениев, фак. Номер: 61773*

Февруари, 2017

София

## Съдържание

1	Описание	2
2	Структура на проекта	4
3	Графичен потребителски интерфейс	3
4	Използвани технологии и алгоритми	3
5	Използвана литература	3

## 1 Описание

Проектът BankCardTokenization има за цел да подобри защитата на банковите карти, чрез токенизация.

Системата има архитектура клиент-сървър като са изпълнени следните качествени характеристики: издръжливост, сигурност, ефективност, използваемост. Системата е изградена на модули, което подпомага нейната преносимост. Реализирана е чрез многонижкова архитектура, позволяваща свързването на повече от един клиент към сървъра, като сървърът изпраща съобщения на всеки клиент поотделно(Multicasting). Комуникацията между сървър и клиент се осъществява чрез request-response messaging pattern като се използват BinaryReader и BinaryWriter за двоични съобщения.

Приложението позволява на клиентите възможност за регистрация и за вход чрез въвеждане на потребителско име и парола. Клиентът трябва да се е логнал в системата, за да пристъпи към основната част на приложението. Ако той не е попълнил коректно данните, изкача съответно съобщение за грешка. Ако данните са коректни, те се записват в файл users.xml и допускат потребителя към възможността за токенизация на банкова карта и достъпване на банкова карта по даден токен в зависимост от правата на потребителя. Една банкова карта може да има няколко токена.

Формата дава право на клиента да въведе валидна карта, за да може сървъра да му върне токен, като той трябва да изпълнява няколко изисквания, описани по-надолу в документацията.

Ако картата не е валидна изписва съобщение за грешка, а в противен случай връща токен по определени критерии, описани по-надолу в документацията.

Картите и токените се записват в файл bank\_cards.xml.

Клиентът има възможност и за въвеждане на токен и връщане на банкова карта при правилно въведен токен. Когато се въведе токен, сървърът проверява за съществуващ такъв и връща банковата карта съответстваща на въведения токен. В противен случай, показва съобщение за грешка. Потребителят може да излезе от системата чрез logout механизъм.

Всичките действия на клиентите се показват като съобщение в графичния интерфейс на сървъра.

Сървърът има две основни функционалности:

- 1) Изваждане на списък от банковите карти и токените, сортирани по номер на банковата карта;
- 2) Изваждане на списък от банковите карти и токените, сортирани по токените;

Те се достъпват чрез натискане на бутон. При натискане на един от тях се отваря диалогов прозорец за записване на съответните данни, които се очаква да бъдат запазени в .txt файл.

Форматът на получения файл е следният:

*Bank Card < - -> Token.*

## 2 Структура на проекта

Проектът се състои от три (основни) части:

**BankCardTokenization.Server** - WPF приложение (Сървър):

- *Objects*
  - BankCard.cs
    - *private string cardNumber* – номер на картата
    - *private List<Tokens> tokens* – лист от токени
  - User.cs
    - *private string username* – никнейм на потребителя
    - *private string password* – парола на потребителя
    - *public string PasswordSalt { get; set; }* – сол на паролата
    - *public UserRights Rights { get; set; }* – правата на потребителя
  - Token.cs
    - *private string id* – съдържанието на токена
    - *private string user* – потребителят, генерирал токена
- BankCardManager.cs
  - Статичен клас, обработващ информация, свързана с банковите карти(генериране на токени и валидиране на номер на карта).
    - *private static string Random rand = new Random()* – инстанция от Random класа, използвана за генериране на „случайни“ числа
    - *public static string GenerateToken(string cardNumber)* – метод, който проверява първо дали номерът на картата е валиден, след което генерира токен като използва последните 4 цифри на номера на картата в токена.
    - *private static string CreateToken(string cardNumber)* – метод, който генерира останалите(първите) 12 цифри на токена на случаен принцип като първо генерира първата цифра, която трябва да е различна от 3,4,5 или 6 и сборът на цифрите на

токена да не е кратен на 10, както и да са различни от тези на номера на банковата карта

- *private static int GenerateRandomDigit(Random rand)* – генерира случайна цифра
- *private static bool IsBankCardNumberValid(string cardNumber)* – метод, който проверява дали е валиден номерата на картата
- *private static bool TestLuhn(string cardNumber)* – валидиране на картата чрез Luhn Algorithm
- *private static bool IsFirstDigitCorrect(int digit)* – проверява дали първата цифра е 3, 4, 5 или 6

- ClientProcessor.cs

– Клас, обработващ заявките на клиента чрез двоични съобщения.

- *private Socket clientSocket* – сокет за клиента (връзката с клиента)
- *private NetworkStream networkStream* – мрежата за клиента
- *private BinaryReader reader* – двоичен четец за клиента
- *private BinaryWriter writer* – двоичен писател за клиента
- *private User user* – настоящия ауторизиран клиент
- *private List<User> Users* – референция към списъка с клиенти
- *private List<BankCard> Cards* – референция към всички банкови карди и токени
- *private Action<string> ProcessMessage* – делегат за обработка на съобщенията
- *private Action<string> ProcessError* – делегат за обработка на грешките
- *public ClientProcessor(Action<string> processMessage, Action<string> processError, List<User> users, List<BankCard> cards)* - конструктор, инициализиращ инстанцията на класа
- *public void ProcessClient(object socket)* – метод, който обработва заявките на клиента в зависимост от правата му
- *private User AuthenticateUser()* – метод за ауторизация на потребителя

- *private User LoginUser()* – метод, който обработва заявките на клиента за логване в системата. След правилно попълване на данните, клиентът може да пристъпи към основната част на системата, в противен случай той бива подканен отново да се оторизира с коректни данни
  - *private User RegisterUser()* - метод, който обработва заявките на клиента за регистрация в системата. След правилно попълване на данните, клиентът може да пристъпи към логване в системата със въведените данни. Паролата на потребителя се криптира и се записва криптирана в списъка, като се добавя и сол, която се запазва също. Използва се *SHA256*.
  - *private bool ManageRequest()* – проверява вида на заявката, чрез изброяването *Operation* и регулира достъпа до функционалностите на системата
  - *private void GenerateToken()* – метод за генериране на токени след валидация на банкова карта. Използва се шаблона “Retry” при генерирането на токен за постигане на отказоустойчивост
  - *private bool TokenAlreadyInUse(string token)* – проверява дали съществува регистриран такъв токен в системата
  - *private void RequestCardNumber()* – метод който изпраща номера на банкова карта по получен токен чрез двоични съобщения
- **Server.cs**
    - Основният клас на сървърната част на приложението, където се стартира и инициализира сървър. Чака се свързването на клиенти към сървър на локалната мрежа по зададен порт като за всеки клиент се пуска нова нишка(multicasting).
      - *private Thread ServerThread { get ; set }* – нишката на сървър

- *private Action<String> ProcessMessage { get; set; }* – делегат за обработка на съобщенията
- *private Action<String> ProcessErrors { get; set; }* – делегат за обработка на грешките
- *private List<Socket> Connections { get; set; }* – всичките свързвания на клиентите
- *public List<Users> Users* – списък от всички потребители
- *public List<BankCard> BankCards* – списък от всички банкови карти с токените им
- *private XMLProcessor xmlProcessor* – използва се за работа с XML файлове
- *public Server(Action<string> processMessage, Action<string> processError)* – конструктор за инициализация на инстанцията на сървъра и зареждане на информацията от xml файловете
- *public void InitializeServer()* – метод, който стартира нишката на сървъра
- *public void RunServer ()* – метод, който подкарва сървъра да чака за заявки за връзка с клиенти като за всеки нов клиент пуска нова нишка(multicast)
- *private void LoadData()* – метод, който зарежда XML файловете(users.xml и bank\_cards.xml) и пълни списъците с потребителите и банковите карти с токените им
- *private void SaveData()* – метод, който запазва информацията от списъците с потребителите и банковите карти с токените им в XML файлове(users.xml и bank\_cards.xml)
- *public void ExportByBankCard()* – метод , който изкарва банковите карти с токените им, сортирани по номер на картата като подканва потребителя да ги запише в файл като се отваря диалогов прозорец
- *public void ExportByToken()* – метод , който изкарва банковите карти с токените им, сортирани по токените на



картата като подканва потребителя да ги запише в файл като се отваря диалогов прозорец

- XMLProcessor.cs

– Клас, който сериализира и десериализира данните, обработвани в системата

- *public void LoadXml<T>(Type type, string filePath, ref T data) where T : new()* – метод, който десериализира данните от XML файл и ги записва в съответния списък, подаден като аргумент.
- *public void SaveXml<T>(Type type, string filePath, List<T> data)* - метод, който сериализира данните от списъка и ги записва в съответния XML файл.

- MainWindow.xaml

– UI на сървъра

- MainWindow.xaml.cs

- Клас, който обработва събитията, провокирани от клиента и потребителя

- *private Server Server { get; set; }* – свойство за достъпване на сървъра
- *public MainWindow()* – конструктор, инициализиращ компонентите на UI и сървъра
- *private void DisplayMessage(string message)* – чака за съобщения и при такива ги изписва в *TextBlock* в графичния интерфейс на сървъра като показва времето, в което се е получило съобщението
- *private void DisplayError(string errorDetails)* – чака за грешки и при такива ги показва в *MessageBox*
- *private void btnExportByToken\_Click(object sender, RoutedEventArgs e)* – при натискане на бутон “Export sorted by

token” в UI се извиква функцията на сървъра за експортиране на банкови карти, подредени по Токени

- *private void btnExportByCardNumber\_Click(object sender, RoutedEventArgs e)* - при натискане на бутон “Export sorted by card number” в UI се извиква функцията на сървъра за експортиране на банкови карти, подредени по номер на банкова карта
- *private void txtMessages\_TextChanged(object sender, TextChangedEventArgs e)* – метод, който обработва събитието *TextChanged* на *TextBlock*-а в UI на сървъра като при всяка промяна(показване на ново съобщение) се скролва автоматично до последното съобщение
- *protected override void OnClosing(CancelEventArgs e)* – предефиниране на метода *OnClosing* като добавяме функционалността при затваряне на приложението данните да се записват в файлове като се извиква функцията на сървъра *SaveData()*

### **BankCardTokenization.Client** - WPF приложение (Клиент):

- *UserControls*
  - *Delegates.cs*
    - *public delegate void RegisterDelegate(string username, string password, UserRights rights);*
    - *public delegate void LoginDelegate(string username, string password);*
    - *public delegate void RequestGenerateTokenDelegate(string cardNumber);*
    - *public delegate void LogoutDelegate();*
  - *LoginUserControl.xaml* – UI на Login UC
  - *LoginUserControl.xaml.cs*
    - *public LoginDelegate ProcessLogin { get; set; }* – свойство на делегата за обработка на логването в системата

- *public LoginUserControl()* – конструктор, който инициализира компонентите на графичния интерфейс
- *private void btnLogin\_Click(object sender, RoutedEventArgs e)* – метод, за обработка на кликване върху бутона Login като се валидират данните първо
- *private void KeyDownHandler(object sender, KeyEventArgs e)* – добавена възможност за логване чрез натискане на Enter бутона на клавиатурата(има само един бутон „Login“ в UC)
- *private bool ValidateUserInput()* – метод, който проверява дали са попълнени текстовите полета. Ако не са, изкача MessageBox с подходящо съобщение.
- *private void ShowRequiredWarning(Control control, string message)* – метод, който показва MessageBox при опит за вписване в системата при непълнени текстови полета(Username или Password)
- RegisterUserControl.xaml – UI на Register UC
- RegisterUserControl.xaml.cs
  - *public RegisterDelegate ProcessRegister { get; set; }* - свойство на делегата за обработка на логването в системата
  - *public RegisterUserControl()* – конструктор, който инициализира компонентите на графичния интерфейс и добавя артикули, съответстващи на правата на възможните права на потребителите, към Checkbox-а на графичния интерфейс
  - *private void btnRegister\_Click(object sender, RoutedEventArgs e)* – метод, за обработка на кликване върху бутона Register като се валидират данните първо
  - *private bool ValideUserInput()* – метод, който валидира данните, въведени в текстовите полета и checkbox-а във формата за регистрация. Те трябва да отговарят на следните изисквания:

- Username: (чрез регулярен израз)
  - Трябва да е между 6 и 30 символа
  - Може да съдържа само букви, числа, точка и долна черта
  - Точката и долната черта не могат да бъдат една до друга
  - Не може да започва или да завършва с точка или долна черта
- Password:
  - Не трябва да е празно полето
  - Трябва да е поне 6 символа
- Confirm Password:
  - Трябва да съвпада с Password
- Rights Checkbox:
  - Трябва да е избрано нещо
- *private void ShowRequiredWarning(Control control, string message)* - метод, който показва *MessageBox* при опит за регистриране в системата при неправилно попълнени данни
- RequestRegisterTokenUserControl.xaml – UI на основната част на клиент приложението
- RequestRegisterTokenUserControl.xaml.cs
  - *public RequestGenerateTokenDelegate ProcessGenerateToken { get; set; }* - свойство на делегата за обработка на генерирането на токен
  - *public RequestGenerateTokenDelegate ProcessRequestBankNumber { get; set; }* - свойство на делегата за обработка на заявка за достъп до номер на банкова карта
  - *public LogoutDelegate ProcessLogout { get; set; }* - свойство на делегата за обработка на отписването от системата

- *public RequestRegisterTokenUserControl()* - конструктор, който инициализира компонентите на графичния интерфейс
- *private void btnGetCardNumber\_Click(object sender, RoutedEventArgs e)* - метод, за обработка на кликване върху бутона "Get Card Number"
- *private void btnGenerateToken\_Click(object sender, RoutedEventArgs e)* - метод, за обработка на кликване върху бутона "Generate Token"
- *private void btnLogout\_Click(object sender, RoutedEventArgs e)* - метод, за обработка на кликване върху бутона "Logout"
- Client.cs
  - Основният клас на клиентското приложение, служещ за обработка на събития, провокирани от клиента и осъществяване на комуникация със сървъра
    - *private TcpClient client { get; set; }* – свойство за връзка със сървъра
    - *public Action<string> ProcessMessage { get; set; }* – делегат за обработка на съобщения
    - *public Action<string> ProcessError { get; set; }* – делегат за обработка на грешки
    - *private NetworkStream NetworkStream { get; set; }* – мрежата към сървъра
    - *private BinaryReader reader { get; set; }* – двоичен четец на клиента
    - *private BinaryWriter writer { get; set; }* – двоичен писател на клиента
    - *public Client(Action<string> processMessage, Action<string> processError)* – конструктор, инициализиращ клиента и делегатите за обработка на съобщения и грешки

- *private void InitializeClient()* – метод, който инициализира клиента като го свързва към сървъра и го подготвя за изпращане и получаване на двоични съобщения по мрежата
- *internal void RequestCardNumber(string token, MaskedTextBox textBoxResponse)* – метод, който изпраща заявка към сървъра за достъп до банкова карта по даден токен
- *internal void GenerateToken(string cardNumber, MaskedTextBox textBoxResponse)* – метод, който изпраща заявка към сървъра за генериране на токен на дадена банкова карта
- *public bool Register(string username, string password, UserRights rights)* – изпращане на заявка за регистрация до сървъра
- *public bool Login(string username, string password)* – изпращане на заявка към сървъра за логване в системата
- *public bool Logout()* – изпращане на заявка към сървъра за отписване от системата
- *public void Dispose()* – метод, който изпраща съобщение на сървъра, че клиента е затворил връзката и спира нишката на клиента.
- MainWindow.xaml
  - UI на клиент приложението
- MainWindow.xaml.cs
  - Клас, който обработва събитията, провокирани от потребителя
  - *private RequestRegisterTokenUserControl ucRequestRegisterToken* – инстанция на *RequestRegisterTokenUserControl*
  - *private Client Client { get; set; }* – свойство за достъпване на клиента
  - *public MainWindow()* – конструктор, инициализиращ клиента, UC за основната част на системата (*RequestRegisterTokenUserControl*) и делегатите в отделните User controls като „ги връзва“ за съответните методи, достъпващи съответната функционалност на клиента

- *private void RequestCardNumber(string token)* – метод, за показване на номер на банкова карта по даден токен
- *private void GenerateToken(string bankCard)* - метод, за генериране на токен на дадена банкова карта
- *private void LoginUser(string username, string password)* – метод, който служи за логване в системата. При успешно такова, потребителя достъпва основната част на приложението
- *private void LogoutUser()* – при успешно излизане от системата, потребителя бива върнат „таба“ за логване
- *private void RegisterUser(string username, string password, UserRights rights)* – при успешно регистриране, потребителя бива изпратен до логин формата като автоматично са попълнени неговите данни от регистрацията и формата за регистрация бива „изчистена“
- *private void DisplayMessage(string message)* – чака за съобщения и при такива ги изписва в TextBlock в графичния интерфейс на сървъра като показва времето, в което се е получило съобщението
- *private void DisplayError(string errorDetails)* – чака за грешки и при такива ги показва в MessageBox
- *protected override void OnClosing(CancelEventArgs e)* – предефиниране на метода *OnClosing* като при затваряне на приложението се извиква метода на клиента за приключване на връзката към сървъра.

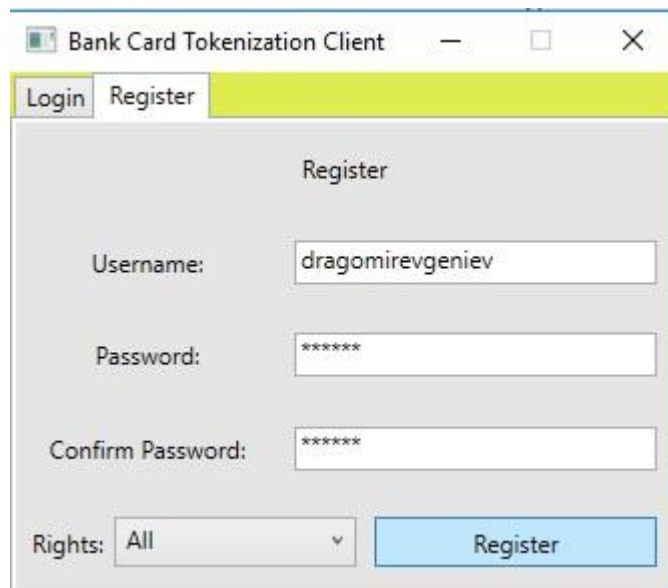
#### **BankCardTokenization.Common** – Class library:

- Constants.cs
  - Статичен клас, съдържащ различни константи, използвани в системата.
- CryptographyManager.cs

- Статичен клас, служещ за криптиране на паролата. Използва библиотеката System.Security.Cryptography. Съдържа следните методи:
  - *public static string GenerateSalt()* – генерира сол на паролата за всеки потребител;
  - *public static string GenerateSHA256Hash()* – хешира паролата на всеки потребител с добавената сол.
- Enumerations.cs
  - Съдържа две изброявания:
    - *public enum Operation* – операциите в системата;
    - *public enum UserRights* – различните права на потребителите.

### 3 Графичен потребителски интерфейс

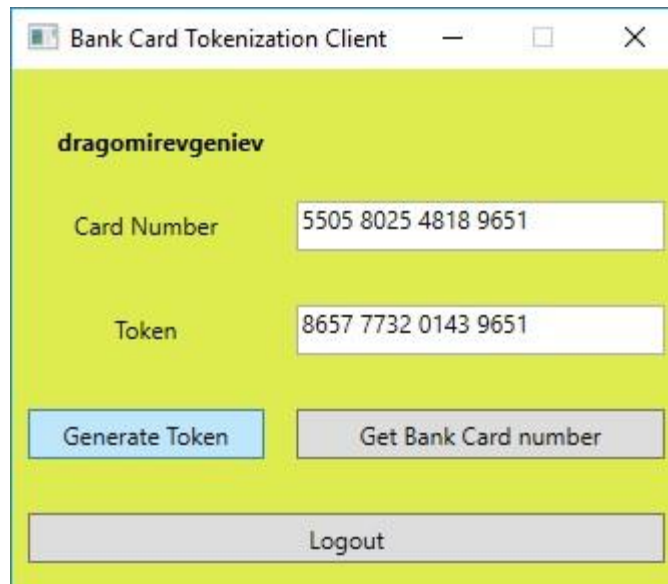
- Клиент



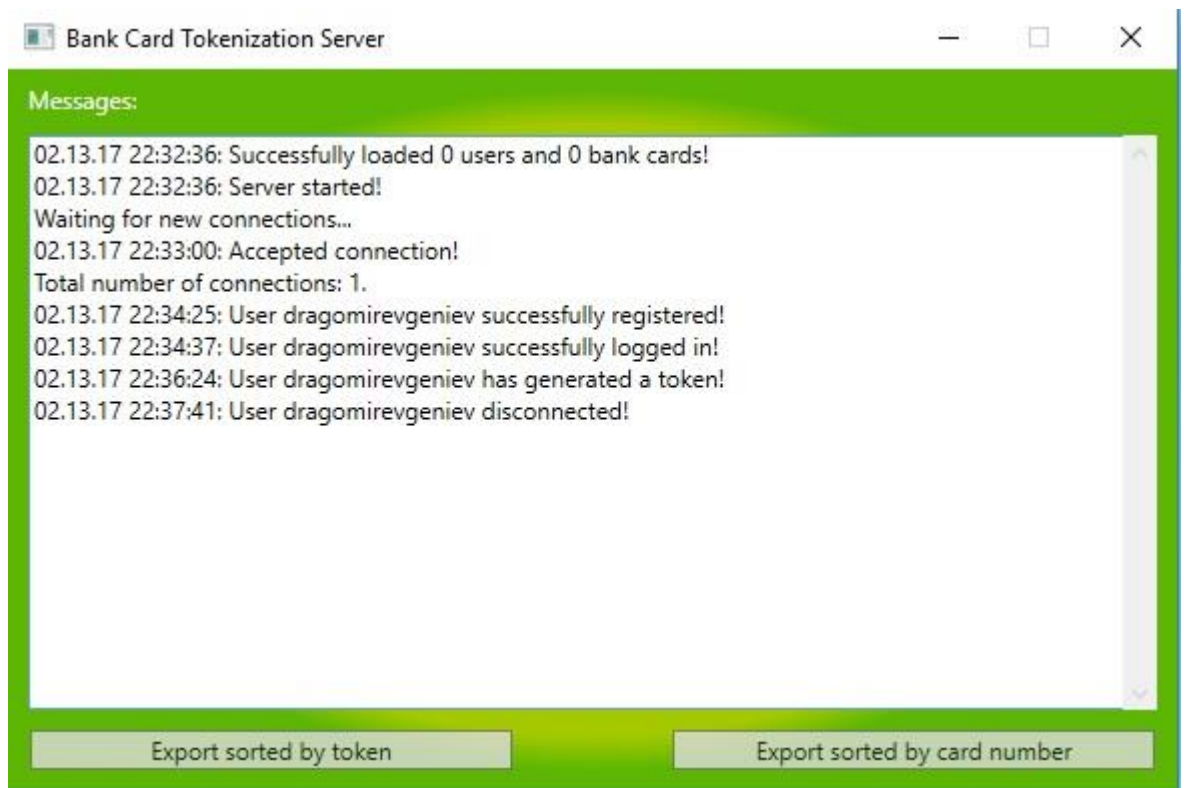
The screenshot shows a Windows application window titled "Bank Card Tokenization Client". It has two tabs: "Login" and "Register", with the "Register" tab currently selected and highlighted in yellow. The "Register" form contains the following fields and controls:

- Username:** A text input field containing the text "dragomirevgeniev".
- Password:** A password input field containing six asterisks "\*\*\*\*\*".
- Confirm Password:** A password input field containing six asterisks "\*\*\*\*\*".
- Rights:** A dropdown menu currently set to "All".
- Register Button:** A blue button labeled "Register" located at the bottom right of the form.





- **Сървър**



## 4 Използвани технологии и алгоритми

- Client-Server Model
- Request-response messaging pattern
- Multithreading
- Retry Pattern
- Password Encryption with salt
- Tokenization
- Luhn Algorithm
- Regular Expressions
- OOP principles
- C# LINQ
- Lambda Expressions
- XML Serializer
- WPF Xceed Toolkit

Системата е написана на програмния език C# в среда за разработка Visual Studio Enterprise 2015.

Валидни номера на банкови карти(за тестване):

### ***Visa***

- 4532079965620928
- 4716160325123494
- 4556925064527151

### ***Mastercard***

- 5339934648955525
- 5505802548189651
- 5208503505486052

## 5 Използвани източници

- Н. М. Deitel, Р. J. Deitel, “Visual C# 2012 How to Program”, 5th ed., Prentice Hall 2013, ISBN 0-13-337933-7 ISBN-13: 978-0-13-337933-4
- Упражнение #15 на курса (Lab #15)
- <https://www.codeproject.com/articles/1415/introduction-to-tcp-client-server-in-c>
- [https://en.wikipedia.org/wiki/Multithreading\\_\(computer\\_architecture\)](https://en.wikipedia.org/wiki/Multithreading_(computer_architecture))
- <http://csharp.net-informations.com/communications/csharp-multi-threaded-server-socket.htm>
- <http://www.mikeadev.net/2012/07/multi-threaded-tcp-server-in-csharp/>
- <https://www.youtube.com/watch?v=zAn-ZbJqS90>
- <https://www.wpftutorial.net/>
- <http://www.getcreditcardnumbers.com/>