**ПРЕДМЕТ: РАЗРАБОТКА НА СОФТУЕР**

***2021-2022***

**Тема: “Приложение, асистент за менажиране на ИТ компании.”**

**Vacation-Manager**

**==================================================================**

*Автори:*

*Калоян Динков Динев 11В*

Съдържание

[1 Въведение 2](#_Toc158534844)

[2 Цели и обхват на софтуерното приложение 3](#_Toc1433353056)

[3 Анализ на решението 3](#_Toc1139737427)

[3.1 Потребителски изисквания и работен процес 3](#_Toc996156012)

[3.2 Примерен потребителски интерфейс 4](#_Toc173240027)

[3.3 Диаграми на анализа 4](#_Toc1250164885)

[3.4 Модел на съдържанието / данните 4](#_Toc974966061)

[4 Дизайн 4](#_Toc1966650307)

[5 Тестване 5](#_Toc423973038)

[6 Заключение и възможно бъдещо развитие 5](#_Toc2029038394)

[7 Използвани литературни източници и Уеб сайтове 5](#_Toc516475091)

[8 Приложения 6](#_Toc219134841)

# Цели и обхват на софтуерното приложение

## Цел

## Обхват

# Анализ на решението

## Потребителски изисквания и работен процес

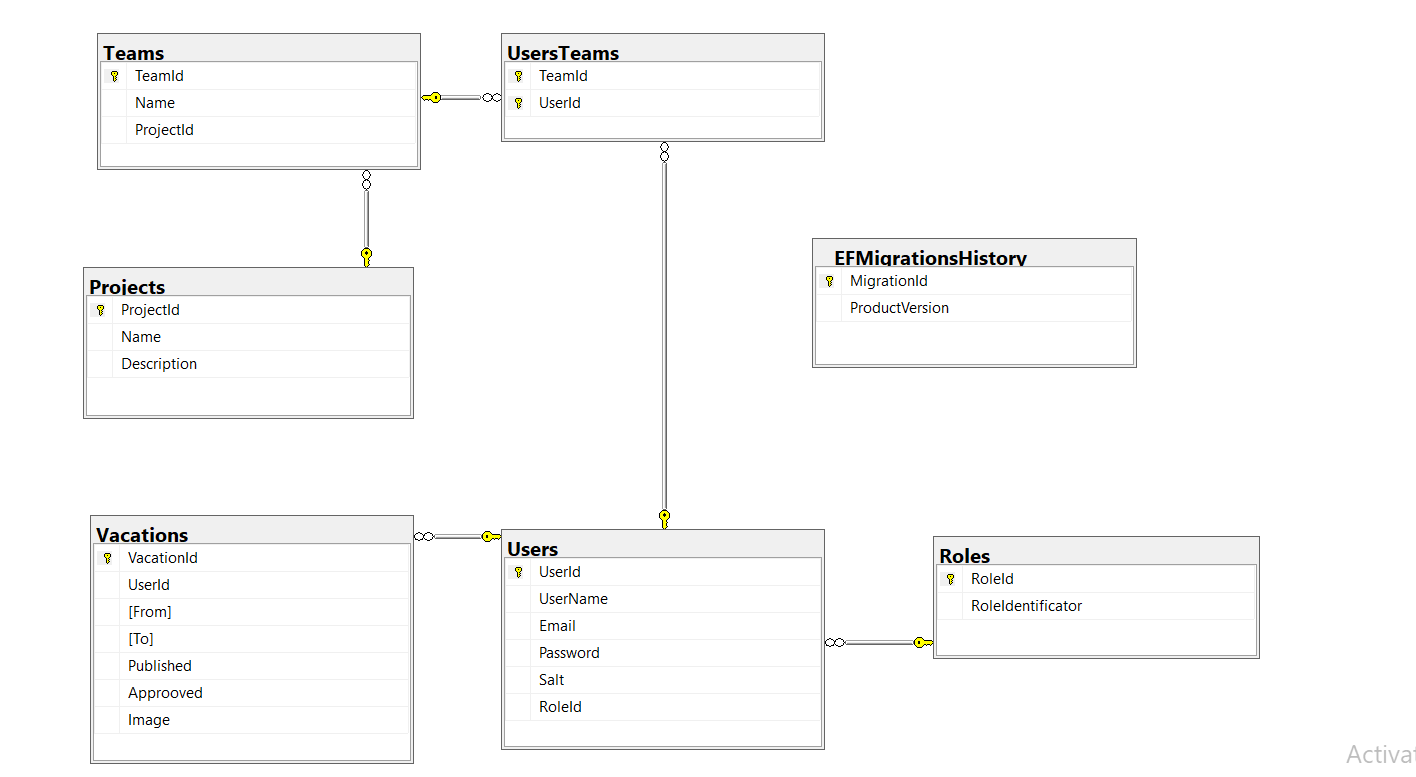
## Примерен потребителски интерфейс

## Диаграми на анализа

### Диаграма на случаите

### Пояснение към диаграмата

## 2.4 Модел на съдържанието / данните



3.4.1 Пояснение към базата данни

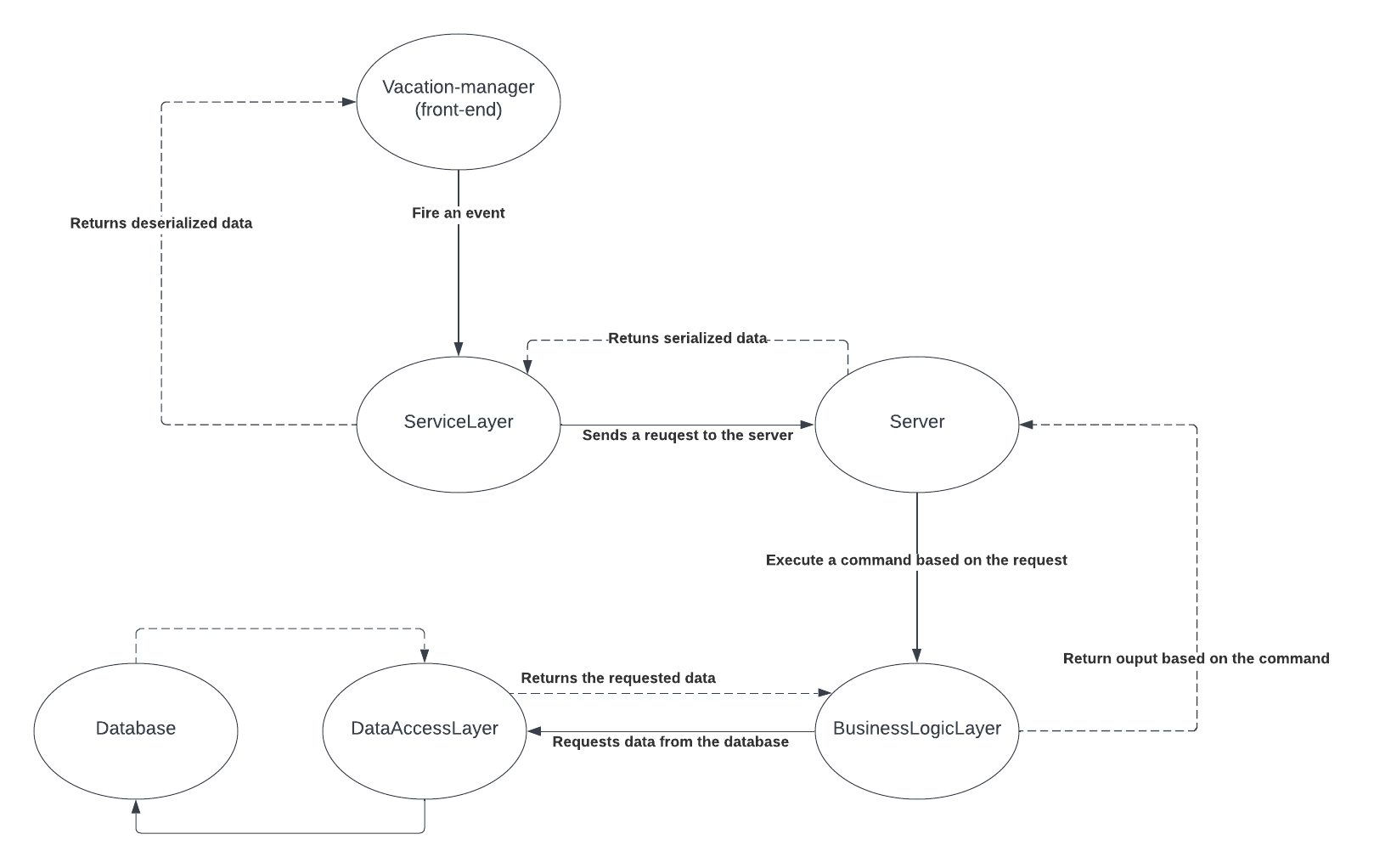
# Дизайн

За реализирането на приложението използвах следните технологии:

1. Visual Studio 2022 – IDE (Програмна среда)
2. Microsoft.Net - C# - Програмен език
3. Windows Presentation Foundation (WPF) - framework за потребителски интерфейс.
4. Entity framework core - framework свързване на кода с бази от данни

WPF не зависи от разделителната способност и използва векторно базирана машина за изобразяване. WPF предоставя изчерпателен набор от функции за разработка на приложения, които включват Extensible Application Markup Language (XAML), контроли, обвързване на данни, оформление, 2D и 3D графики, анимация, стилове, шаблони, документи, медии, текст и типография. WPF е част от .NET, така че можете да създавате приложения, които включват други елементи на .NET API.

Диаграма на решението:



Пояснения по диаграмата:

Работния процес, при който една команда бива изпълнена, е следния:

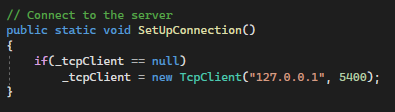
1. Потребителя си взаимодейства с потребителския интерфейс
2. Потребителския интерфейс изпълнява съответното събитие
3. Събитието извиква метод от ServiceLayer-а съответстващ на поисканата команда
4. ServiceLayer-ът изпраща заявка към сървъра
5. Сървърът, базирано на заявката, изпълнява команда от BusinessLogicLayer-а
6. BusinessLogicLayer-ът прави заявка към базата от данни
7. Базата от данни връща съответния отговор, базирано на заявката
8. BusinessLogicLayer-ът връща съответната информация на сървъра
9. Сървърът връща сериализирана информация на ServiceLayer-а
10. ServiceLayer-ът десериализира информацията и я праща на front-end-а

Пояснения по работния процес:

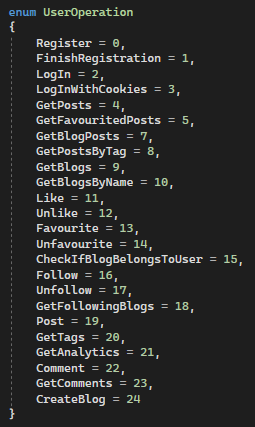
Проектът ще се състои от две части: клиентска и сървърна. Клиентската част ще се състой от front-end-а и ServiceLayer-а. Като ServiceLayer-ът ще бъде този, който си комуникира с сървъра. Сървърът ще се състои от: самия сървър, бизнес логиката, дата достъпващи класове и базата от данни.

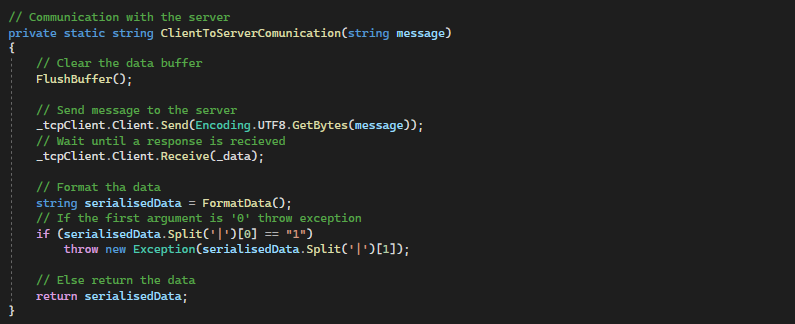
Комуникацията ще се случва over the air. Тоест няма да има нужда двете чати да имат пряка връзка. Нито пък да са на една и съща машина. Сървърът е конструиран така, че да може да приема много потребители на веднъж, позволявайки много потребители да са активни по едно и също време

Как се осъществява връзката между ServiceLayer-а и Сървъра

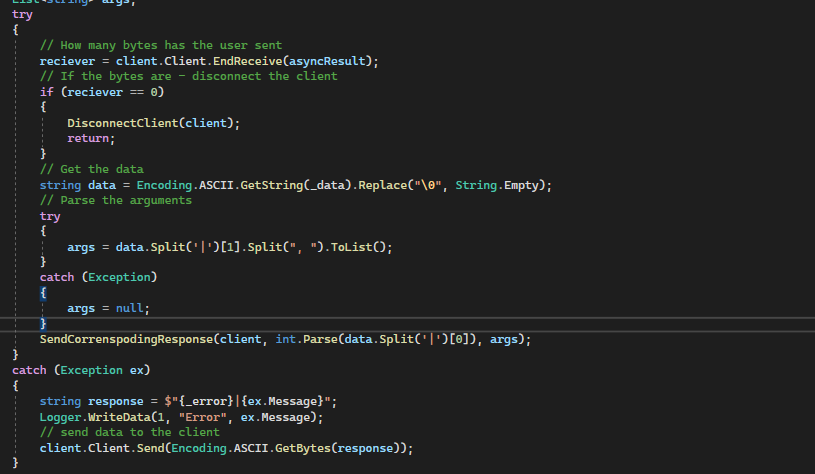
Пот този начин ServiceLayer-ът се свързва към сървъра. На мястото на "127.0.0.1" ще бъде поставено вашия публичен ip адрес. ”5400” е портът, който сървъра използва.

Как се осъществява комуникацията:

С еномератора UserOperations номерираме всяка възможна операция, която потребителя може да извика, чрез потребителския интерфейс. Този еномератор се намира какво в ServiceLayer-а така и в Server-a. Той се използва, за да може Server-a да разбере каква операция да изпълни за дадена заявка от ServiceLayer-а.



Комуникацията се осъществява чрез метода ”ClientToServerComunication” с аргумент съобщението което искаме да пратим на сървъра. Първо изчистваме дата буферът от предишни комуникации, след което пращаме съобщението, конвертирано в байтове, към сървъра. Client.Receive е блокираща операция, тоест програмата няма да продължи докато клиента не получи отговор. След като получим очаквания от нас отговор, конвертираме дата буфера в низ от символи. И накрая се проверява се дали операцията е била успешна, ако не се предизвиква изключение, ако да метода връща низа от символи.



По този начин сървърът получава изпратеното от клиента съобщение. Първо проверява дали пратения брой байтове е 0 и ако е, разскача клиента от сървъра. В противен случай байтовете се конвертират към символен низ. След което се изпълнява метода “SendCorrenspodingResponse”, който извиква кореспондиращия, на изказваната операция от клиента, метод от BusinessLogicLayer-а и връща съобщение на клиента.

Как се формулират съобщенията:

Съобщенията, които сървъра получава се сформират по следния начин: номер на операция|аргументи разделени със запетая. Пример с операция “Register”:

0|admin, [admin@admin.com](mailto:admin@admin.com), Password1!2@

Съобщенията, които сървъра изпраща се сформират по следния начин: дали операцията е успешна|сериализиран отговор. 0 означава, че операцията е успешна, а 1 означава че операцията е неуспешна. При не успешна операция съобщението ще се сформира по следния начин : 1|име на съобщението за грешка.

Сървъра е мултиклиентов, тоест много клиенти могат да се свържат към сървъра. Сървъра създава нова нишка(единицата, която изпълнява код), за всеки отделен потребител. Когато потребителя затвори връзката нишката не се изтрива, а се маркира като свободна.

При всяка заявка сървъра създава нов контекст(единицата, която използваме за направа на заявки към базата). Този контекст се премахва веднъж щом заявката приключи нейното изпълнение.

## Реализация на структура на приложението (n-layer), Разделение на кода според предназначението му

## Организация и код на заявките към база от данни

# Тестване

Тук *само маркирайте какво трябва да включват тестовите случаи* и какви видове тестване предвиждате в реалното изпълнение на проекта, напр. с колко и какви документи, в какви браузъри, с какви приставки, и т.н.

# Заключение и възможно бъдещо развитие

В заключение, обобщете резултатите от работата ви по проекта, както и предимствата и ограничеността на използваните технологии / езици / методи. Укажете какви алтернативи могат да се използват и техните предимства и недостатъци. Опишете каква е използваемостта на подобни решения в практиката и какво бихте предложили като насоки за бъдещо развитие на вашето решение.

# Използвани литературни източници и Уеб сайтове

1. Lucid chart - https://www.lucidchart.com/pages/