БАКАЛАВЪРСКИ ФАКУЛТЕТ

Департамент „Информатика“

Програма „Информационни технологии“

Курсова работа

CITB679 Практика по бизнес информационни системи

Група 6: NET.Core - *Система за сравнение на банкови продукти*

Ръководител: .....

/гл. ас. д-р Лъчезар Томов/

Студенти:.....

/Стоян Лупов ...../

/Мирослав Маринов ...../

/Момчил Димитров...../

Юни 2019 г.

Съдържание

1. Въведение

VashiteKinti предоставя възможност на клиента да намери най-подходящата банка за него, в която да направи съответен депозит. От навигацията в сайта ефективно и бързо може да се избере предпочитания подход за преглед на оферти, изчисляване на най-удачната от тях и извършване на депозит в банка. Потребителите имат възможността да си направят регистрация, което ще им осигури пълен достъп до всички функционалности на сайта.

За прецизното изчисление на най-подходящата оферта, клиентът може да попълни следните показатели: пет основни валути, размер на депозита, период на депозита, за кого е депозита, вида и начина на изплащане на лихвата, довнасяне на суми, възможност са овърдрафт и възможност за кредит. След като сравни наличните оферти ще може да достъпва функцията „Изчисли“, която го препраща към страница с разплащателен план за конкретния депозит при съответната банка.

При желание на клиента за преглед на всички налични видове депозити, сайтът съдържа Каталог. В него те могат да бъдат филтрирани по два различни компонента: „валута“ и „метод за изплащане на лихва“. След филтриране и избор на съответния депозит, на потребителя се предоставя разпащателен план при натискане на бутона изчисли. На вниманието на клиента се представя и списък на банките, чиито услуги предлага VashiteKinti. Платформата разполага с десктоп приложение, което позволява попълване на данни за различни банкови продукти.

1.1 Анализ на риска

В проекта идентифицирахме риск свързан с времето, неободимо за изготвянето на изцяло завършена платформа с всичките му функционалности поради невъзможността да се събере екипа в пълен състав. По този начин срещнахме затруднения в стремежа си да не

лишаваме проекта от неговата цялост. Имахме за цел значително да ускорим процеса изграждайки проекта заедно, а не на части, в спокойна среда за разработка, която да ни осигури нужните условия за постигането на очаквания резултат максималко бързо и ефективно. Последствията от гореспоменатите рискове са предпоставка за по-голяма натовареност и затруднения в пълното изпъление на заданието в изчисления период от време.

2. Разпеделение на задачите

За изпълнението на проекта и максималното да симулиране на реална работна среда решихме, че ще е най-удачно да работим по Scrum методологията, защото в момента тя е една от най-разпространените в индустрията. Броя на участниците в екипа ни отговаря на стандартите за „скръм“ екип. Комбинирахме по няколко длъжности на човек, защото в един добре функциониращ екип има софтуерни инженери, тестери, архитект, дизайнер и scrum master. Това са и назначенията, които бяха разпределени в нашия екип. Още в началото обсъдихме подробно каква ще бъде архитектурата на проекта и какъв да бъде нашия MVP, към който да се стремим.

3. Структура на проекта

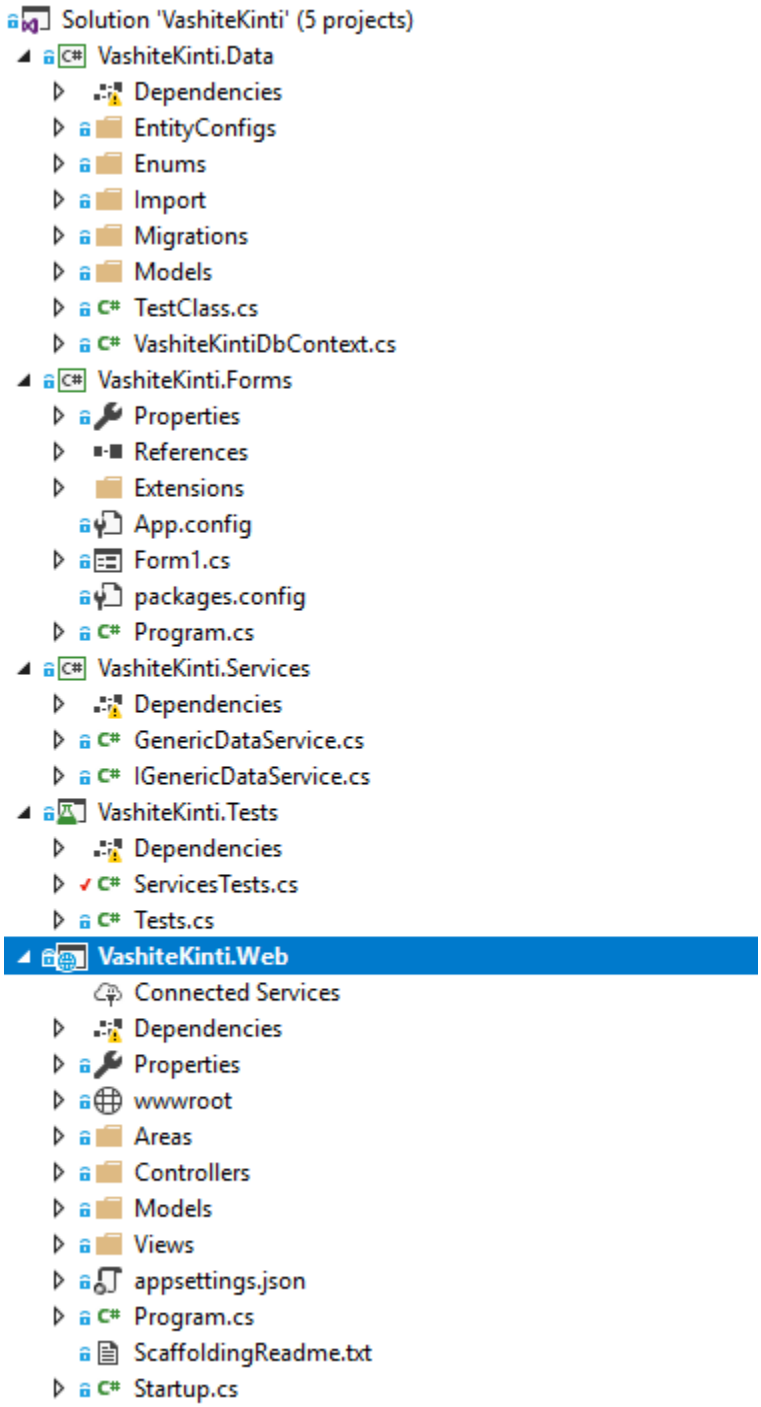
Голяма част от проекта е написана под .NET Core 2.2. Приложението представлява ASP.NET Core MVC уеб апликация с връзка към база данни SQL Server. Базата и приложението общуват посредством ORM Framework – Entity Framework Core.

Фронтенд технологиите включват HTML, CSS, JavaScript (JQuery).

Към проектът се намира второ изпълнимо приложение – Windows desktop апликация написана на WinForms. През нея потребителят може да добавя, редактира и изтрива записи за депозитите в базата от данни.

Библиотеката за Unit тестове, използвана в проекта е Xunit.

Управлението на потребителските акаунти и техните роли се извършва от вградената Identity framework на ASP.NET Core. Нужни бяха няколко промени по настройките й по подразбиране, за да напаснат за нашите нужди. Изискванията за регистрация на потребител бяха променени. Сега потребителите се автентикират с потребителско име, а не с e-mail. Ограниченията по паролите са значително по-малки и лесни. Ролите за потребителите са строго дефинирани.



3.1 VashiteKinti.Data

В тази споделена библиотека се помещават моделите за таблиците в базата от данни, техните конфигурации контекстът за базата, миграциите съставени при промени по схемата на базата и json

файловете свързани с първоначалния сийд на базата с примерни стойности.

Проектът работи по методологията Code First database, където при промяна на някой от моделите или негова конфигурация се прилага миграция на базата.

С пускане на приложението, базата се пълни с първоначални стойности на някои от таблиците. Администраторският акаунт се създава в този момент като информацията за неговите креденшъли се намира в appsettings.json. Това е неправилно, тъй като тази информация се намира в дървото на проекта и е видимо от всеки притежател на сорс кода. Правилният начин е подобни тайни да се помещават в предоставените от Майкрософт User Secrets локално на сървъра. Но понеже това би означавало прехвърляне на user secrets файлове от един разработчик на друг по време на разработване на проекта, решихме че за целите на курсовата работа това е ненужно и ползваме appsettings.json.

Библиотеката VashiteKinti.Data се билдва с таргет версии .netcore2.2 и .net framework 4.6.1 поради нуждата десктоп приложението за обработка на депозите да се линкне към нея.

3.2 VashiteKinti.Services

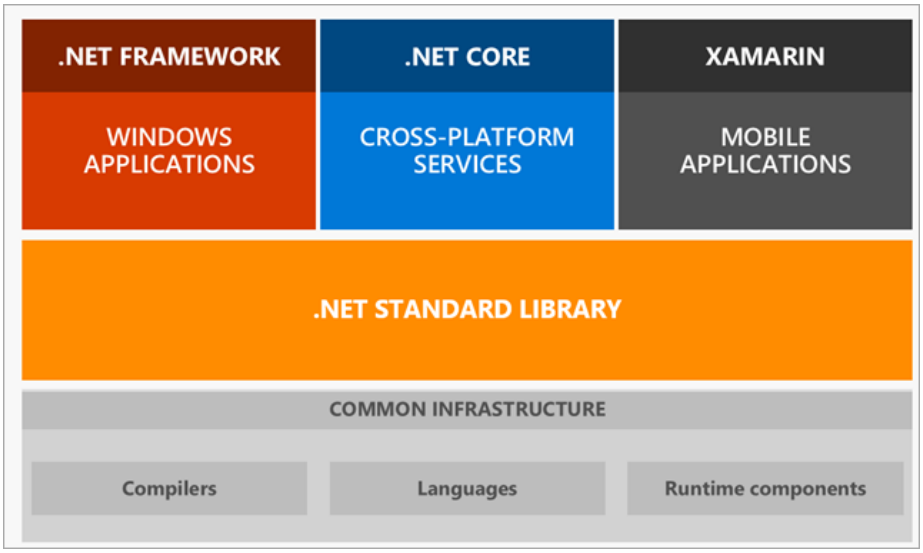
Този проект включва два главни файла. Generic интерфейс служещ за описание на нужните функции за достъпване до неопределена таблица от базата и имплементацията на този интерфейс. Тук са описани функции за извличане, редактиране, добавяне, изтриване и проверка за наличност на елементи в базата.

Тези сървиси за достъп до базата се добавят в IoC контейнерите на главното приложение VashiteKinti.Web, за улеснен достъп до тях и лесен начин за употреба от бизнес логиката.

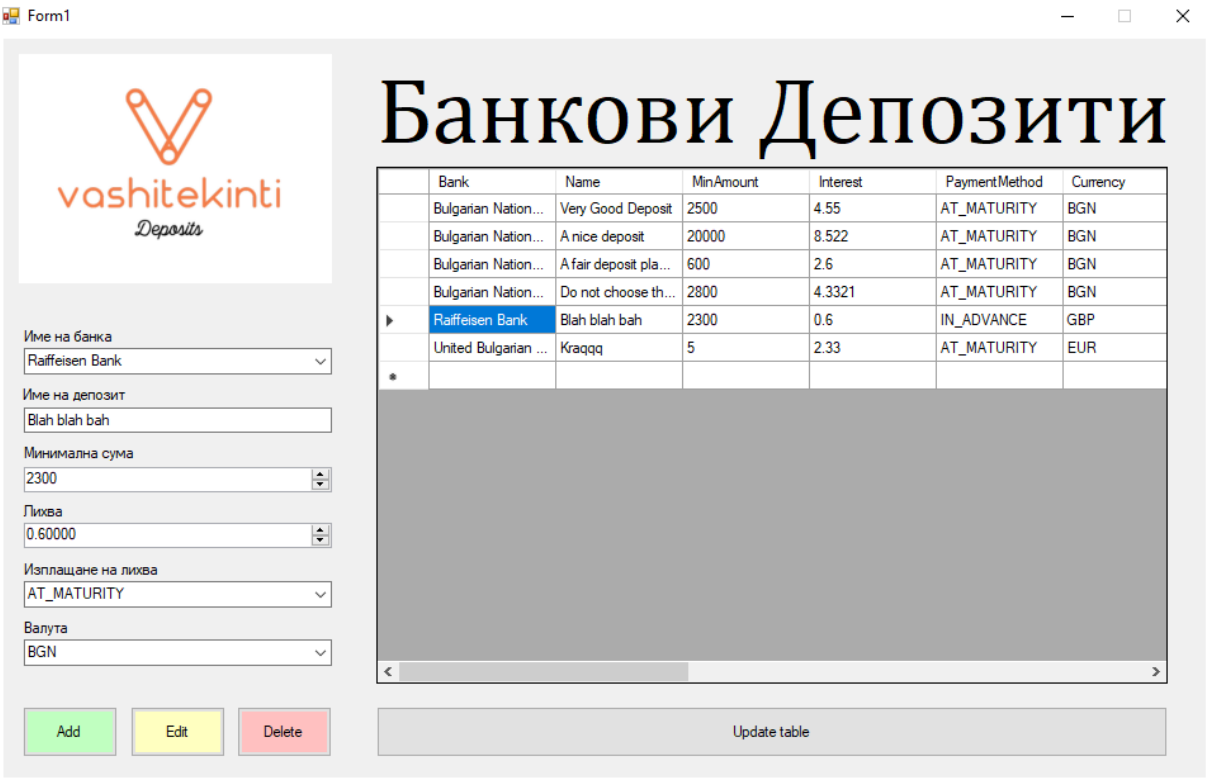
Библиотеката VashiteKinti.Services се билдва с таргет версии .netcore2.2 и .net framework 4.6.1 поради нуждата десктоп приложението за обработка на депозите да се линкне към нея.

3.3 VashiteKinti.Forms

Проектът представлява десктоп приложение на .net framework 4.6.1 (десктоп приложения не са възможни на .NET Core преди версия 3.0, която все още не е излезнала официално). Приложението преизползва библиотеките VashiteKinti.Data и VashiteKinti.Services за да избегне пренаписване на описанието на базата и евентуални промени по нейната схема. Нуждата от приложение на .net framework 4.6.1 да използва библиотеки на .NET Core 2.2 доведе до необходимостта от билдване на въпросните библиотеки на два различни таргет фреймуърка. Това се оказа най-лесния и бърз начин за линкване на двата таргета понеже вторият вариант, а именно билдване на библиотеките към общия .NET Standard фрейъмърк би довело до орязване на доста от функционалностите и предлаганите APIта.



Вече разполагайки с компилиращи се проекти целта беше да се напише приложение за десктоп, което да преизползва доста от логиката по модифициране на базата.



Формата предлага листване на всички запазени в базата депозити (чрез бутона „Update table”). След селекция на определен обект от таблицата, неговата информация е пренесена на полето за модификация от ляво на таблицата. От там предоставените данни могат да бъдат редактирани, записът може да бъде премахнат или да бъде добавен абсолютно нов.

Всичките тези операции се случват след натискане на съответния бутон от трите избора долу вляво. Всяка операция модифицира състоянието на записите в базата.

3.4 VashiteKinti.Web

Главното уеб приложение на проекта. Съставено е от познатата MVC архитекура. Ползва VashiteKinti.Data и VashiteKinti.Services за осъществаване на връзка с базата. В началната точка на приложението Startup.cs се регистрират сървисите в IoT контейнер за по-удобна употреба от контролерите (където се случва бизнес логиката на сървъра). Тук още се осъществява сийдването на базата от данни, съставянето на различните роли за потребителите, конфигурация на вътрешни сървиси и т.н.

В папка Controllers се помещават класовете осъществяващи бизнес логиката на приложението и предаване на данни за визулазиране към изгледите (Views). Във всеки контролер се инжектира посредством конструктура му сървис за връзка с определената нужна таблица от базата данни. Това става с помощта на гореупоменатия IoC контейнер, който за разлика от IoC контейнера на VashiteKinti.Forms (там се ползва SimpleInejctor) тук идва наготово от ASP.Net Core MVC фреймуърка.

В папка Models се намират класовете, които се предават на изгледите като данни за визуализиране. Тях наричаме view models.

В папка Views се помещават всички cshtml файлове, които описват изгледа на страницата с нормален HTML, но примесен със C# код съдържащ лека логика по конструирането на по-динамичен HTML код. Това става с любезното съдействие на Razor технологията предоставена от .NET.

В папка wwwroot се помещават статични файлове за сървъра като ресурси (картинки, звуци, скриптове, CSS файлове и т.н). Тук се намират картинките за логата на всички банки. В по-голям проект бихме използвали външна CDN услуга за съхранение на ресурсни файлове, но

следващата най-добра алтернатива за проект от нашия мащаб се оказва съхранение директно във файловата система на сървъра.

3.5 VashiteKinti.Tests

Този проект съдържа 2 файла. В единия е написана функция, която се използва за достъпване на базата за улеснение при тествнето. За тестовете е използвана In memory database. В другия файл са тествани всички сървиси от проекта VashiteKinti.Services. Тестовете са осъществени като срещу In memory database-а са ползвани GenericDataService<T> като T е депозит, банка или нещо друго.