МЕНИДЖЪР НА НЕДВИЖИМИ ИМОТИ – АРХИТЕКТУРЕН ПРОЕКТ

Проектна група: 5

Автори:

Иван Пиришанчин 471218055

Борислав Крумов 471218065

Валентин Върбанов 471218063

Дата: 27/10/2020

Ревизиран: 10/01/2021

Представяне на документа

Целта на този документ е да опише софтуерната архитектура на проекта „Мениджър на недвижими имоти“. Ще добием обща представа за модулите, от които е изградена системата, но ще пропуснем конкретиката за тяхната реализация. Взаимодействието между тези модули е предмет на софтуерната архитектура.

Този документ обхваща фазата на проектиране, като абстрактно описва фазите на имплементация/програмиране и внедряване.

Актьори

Участниците, заинтересовани от документа са:

- Софтуерни разработчици
- Служители в агенцията за недвижими имоти (ползватели на проекта)

Използвани термини и символи

ASP.NET – технология за разработване на Web приложения на Microsoft

MVC – архитектурен шаблон, целящ разделение на данни, бизнес логика и потребителски интерфейс, използван в ASP.NET

База данни – съвкупност от таблици, които пазят данни (да не се бърка със СУБД)

Entity Framework – връзка обект-релация, библиотека, която позволява да изпълняваме C# код в/у база от данни (вместо суров SQL)

CRUD – Създаване/Четене/Обновяване/Изтриване, основни операции при работа с бази данни.

Db Provider – също може да се срещне и като Db Driver; предоставя адекватен интерфейс между съответен програмен език от високо ниво и източник на данни (data source)

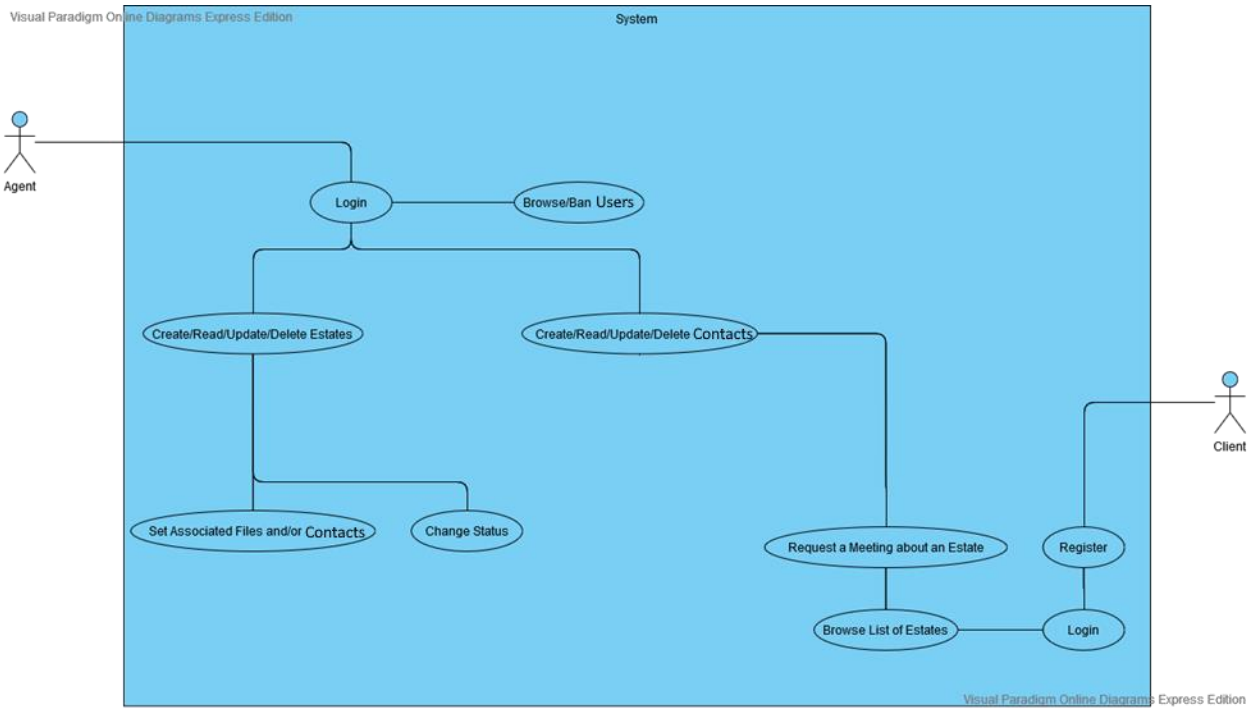
Entity – тип, представящ таблица в базата данни, за употреба от език на високо ниво

Repository Pattern – абстракция (Design Abstraction), която спестява на ползвателя на базата данни детайлите на имплементацията на съответните операции над нея

.NET Framework – предоставя основна функционалност за проекта и позволява работа с езика C#

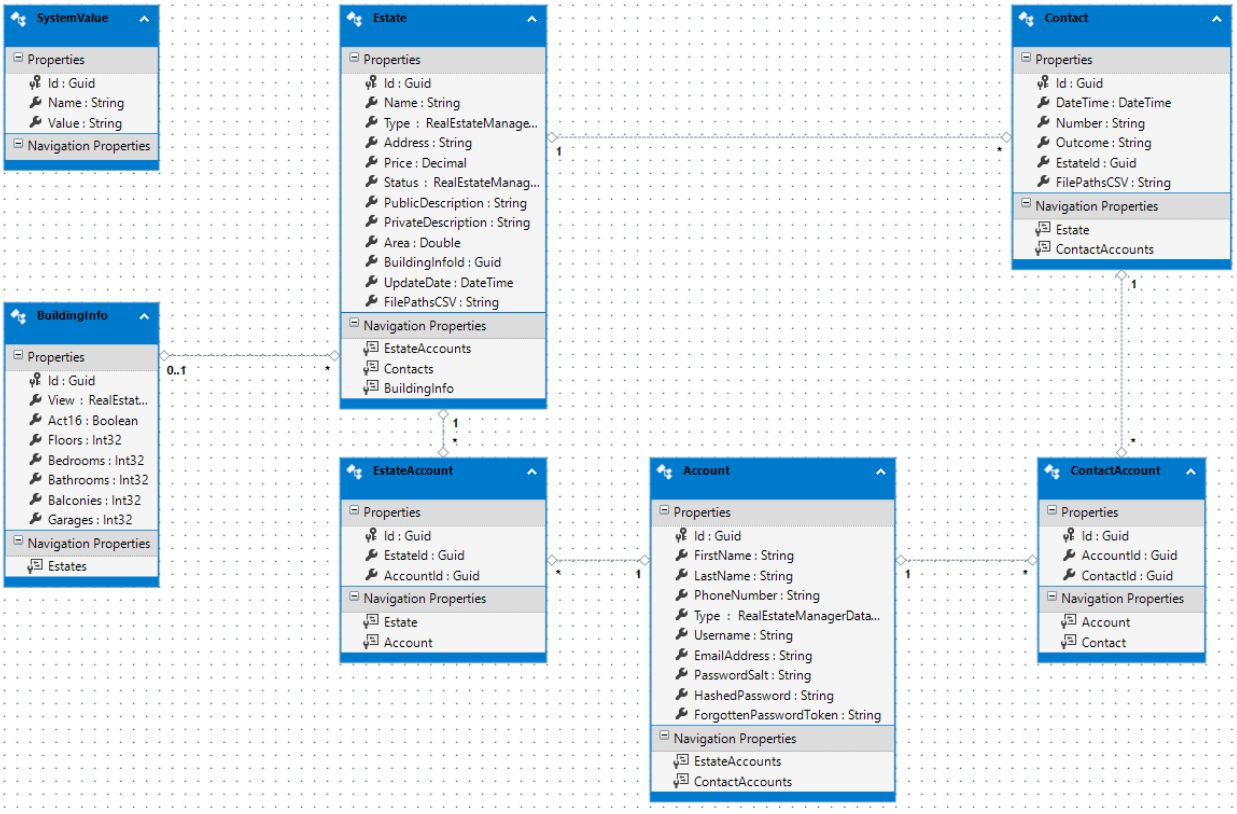
Base Class Library (BCL) – библиотека с множество възможности, предоставяща главни характеристики на C#.

Архитектурен обзор



Диаграмата описва как един брокер (Agent) може да използва системата. Имаме обособена функция за вход и достъп до администраторския панел. Там агентът има достъп до множество от опции, като CRUD на имоти, срещи и файлове.

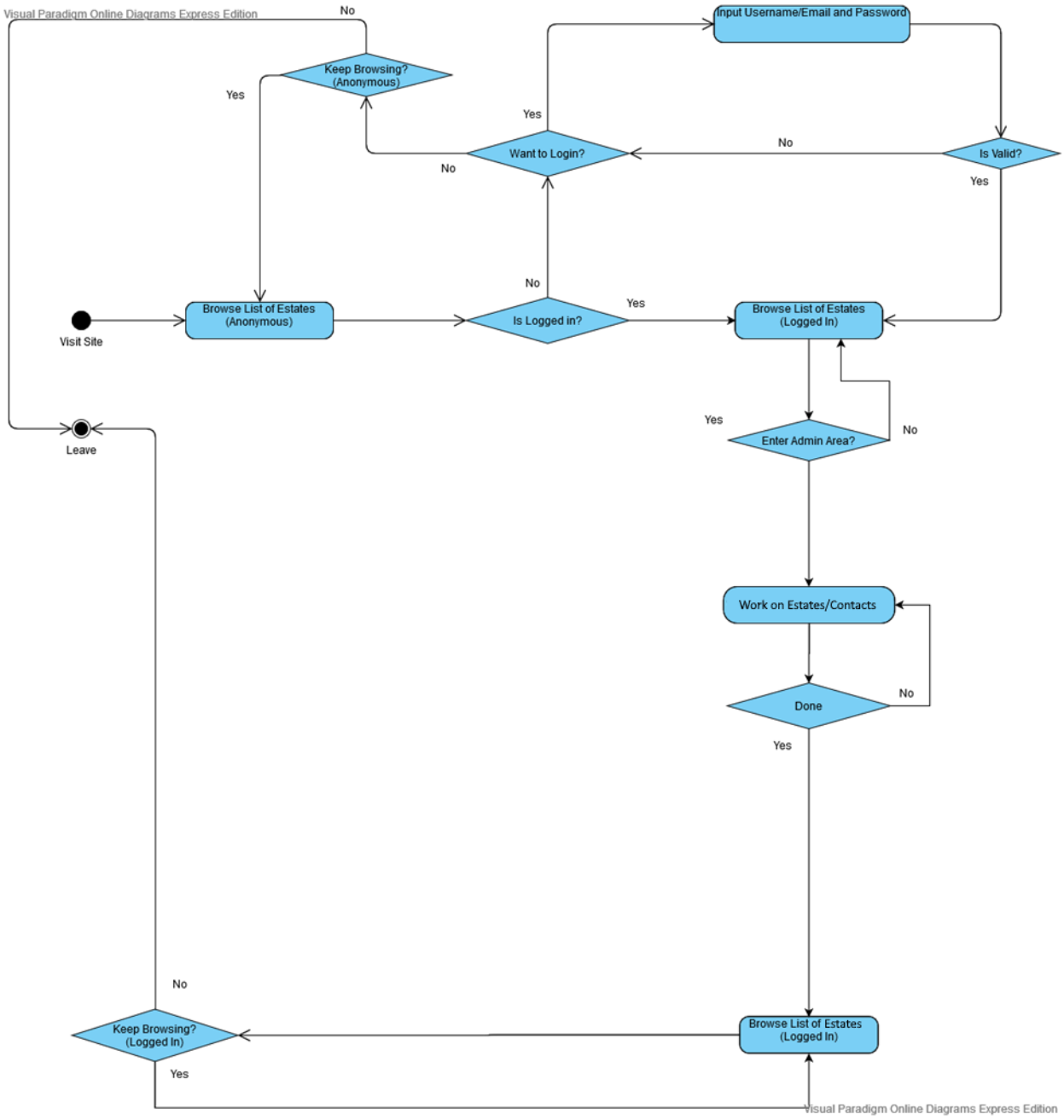
От друга страна купувачът трябва първо да се регистрира, да влезе в акаунта си и тогава ще може да потърси среща с агент за даден имот. Агентът трябва да обработи срещата.



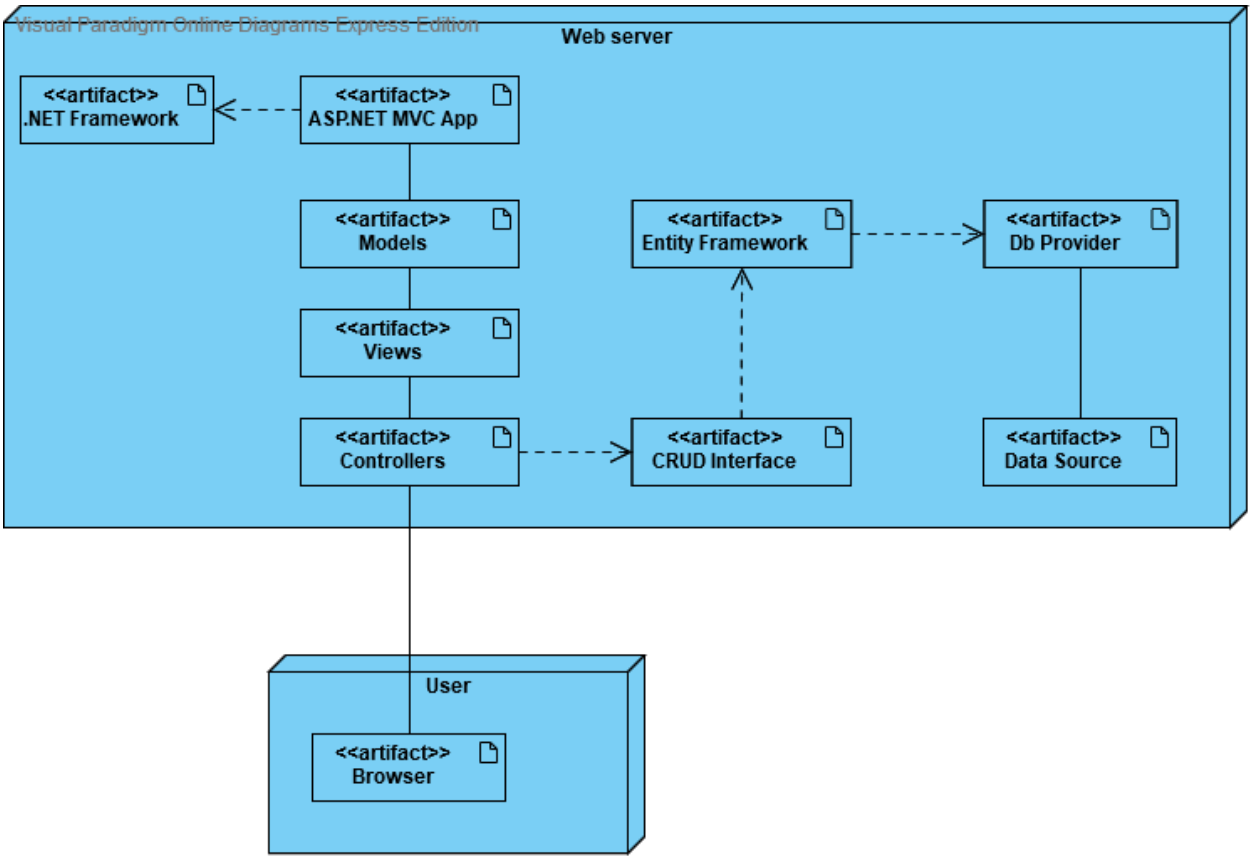
Описана е архитектурата на слоя за данни. От една страна имаме автоматично генерирани класове от Entity Framework – Контекст на базата данни, Entities и DbSet-ове.

За да скрием детайлите на имплементацията, имаме класовете, част от Repository Pattern. Идеята е, че операциите над всеки обект подлежат на различни логически ограничения (logical constraints). Защо трябва ползвателя (контролерите) да знае и да повтаря детайли на имплементацията във всяка операция? Проверки за валидност? За това имаме добре дефинирана абстракция между ползвателя и базата данни, на която ползвателя трябва да подава само абсолютно необходимата информация.

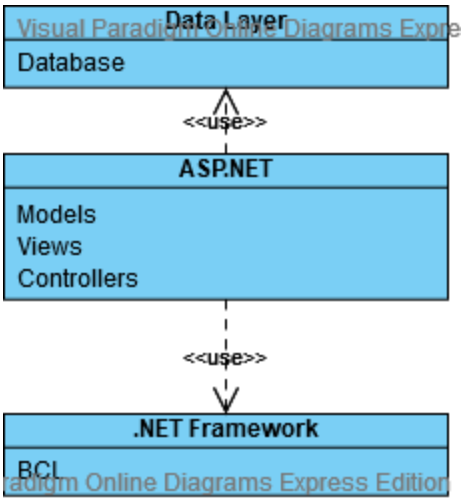
Така елиминираме повтаряемостта и възможността за логическа грешка при подаване на данни.



Диаграмите за активност описват как двата вида потребители на системата могат да работят с нея. Анонимните потребители могат да преглеждат публичния каталог с имоти. След вход в системата те могат отново да гледат каталога с имот, или ако са агенти, могат да влязат в Административния панел и да работят по имоти/срещи/документи.



Описана е архитектурата на системата. Имаме уеб сървър (Windows Server OS), с .NET Framework и IIS. Нашето ASP.NET приложение се хоства на сървъра. Под CRUD Interface се разбира repository pattern-а описан отгоре. Db Provider-а се настройва в проекта и предоставя достъп до източника на данни (База данни).



Вече е ясно, че се оформят няколко слоя:

- слой на данни – база данни, модели
- слой на бизнес логика – .NET Framework с BCL, контролери
- слой на потребителския интерфейс – изгледи

Нефункционални изисквания

Избраната архитектура реализира следните атрибути за качество:

Достъпност:

- Трябва да бъде осигурен бърз и лесен достъп до електронните услуги и те да бъдат промотирани с подходящи навигационни елементи на публичната интернет страница – банери, елементи от главното меню и др.
- Публичните уеб страници на Системата трябва да бъдат проектирани и оптимизирани за ефективно и бързо индексиране от търсещи машини с цел популяризиране сред потребителите и по-добра откриваемост при търсене по ключови думи и фрази.
- В екранните форми на Системата трябва да се използват потребителски бутони с унифициран размер и лесни за разбиране текстове в еднакъв стил.
- Полета, опции от менюта и командни бутони, които не са разрешени конкретно за ролята на влезлия в системата потребител, не трябва да са достъпни за този потребител.
- Всяка екранна форма трябва да има наименование, което да се изписва в горната част на екранната форма. Наименованията трябва да подсказват на потребителя какво е предназначението на формата.
- Полетата за пароли трябва задължително да различават малки и главни букви.

Разширяемост:

Системата трябва да бъде реализирана със софтуерна архитектура, ориентирана към потребителския интерфейс – MVC (Model View Controller).

Това е така, защото при мащабен проект като този е необходимо да има ясно разграничение между данни, бизнес логика и потребителски интерфейс.

Потребителският интерфейс е най-често променящият се софтуер при интерактивните системи. За това е важно да се отделят промените върху интерфейса, така че да не влияят върху останалата част от системата. Потребителите често искат да гледат на системата от различни гледни точки (различни гледки).

Изсивкането е системата да се раздели на 3 основни компонента:

- модел: съдържащ данните на приложението
- изглед: визуализиращ данните

- контролер: обработва заявки към системата с определени данни (от модел) и връща определена страница (от изглед)

Ефективност:

- Отделяне на преизползваемите частични изгледи за вграждането им в различни страници.

Сигурност:

- Удостоверяване на потребителя (User authentication): проверява се дали потребителят е този, за когото се представя. Използват се парола при идентификация.
- Идентифициране на актьори (Identify actors): Идентифицира се източника на връзката със системата. Потребителите се идентифицират чрез потребителските им имена.
- Разрешаване на достъп на актьорите (Authorize actors): Идентифицираният потребител има права да изменя данните или ползва услугите на системата. Контролът на достъп до системата се реализира чрез задаване на роли на потребители или групи.
- Паролите на потребителите се съхраняват в базата данни чрез алгоритъм за хеширане

Използваемост:

- Бутоните за навигация използват различни цветове, когато са селектирани или активни
- Системата предоставя функция за търсене по текст.
- При по-сложни или комплексни операции ще бъдат осигурени упътвания за функционалностите