**Документация**

1. Архитектура

Архитектурата е трислойна. Data, Business и Presentation layers.

За по-добра организация на файловете проекта е изцяло рефакториран спрямо дефолтния MVC 5 проект. Изнесени са DbContext-a и ApplicationUser в съответни отделни проекти. Също, много класове, които by default са в общи файлове са изнесени в отделни. Изчистени са Stylecop грешките.

За инжектиране на dependencies е изпозван Autofac. Конфигурацията му е в „…\Hangman\Source\Web\Hangman.Web\App\_Start\AutofacConfig.cs“.

За мапинг между проектите планирах да изпозвам AutoMapper и съм го конфигурирал, но само на едно място съм изпозвал мапинг и го реализирах със статично пропърти Expression<Func<,>>, което се подава директно във Select функцията на LinqToEntities.

**Data Layer**

За комуникация с базата използвам EntityFramework. Конструирал съм базата чрез Code First Approach. За комуникация с контекста на базата съм приложил Repository Pattern.

Всички модели в базата наследяват BaseModel в който са сложени общи пропъртита, като id, isDeleted, CreatedOn ... и т.н.

1. Имплементация на игровата логика.

Двоумях се дали да пазя текущите игри в базата. Щеше да е по-лесно, но трябваше да се затрупа базата със заявки. Реших текущите игри да се пазят в колекция заредена в оперативната памет. Всички класове отговарящи за това са в папка „Hangman\Source\Web\Hangman.Web\ActiveGames“. Основния клас е ActiveGamesManager.cs, който пази в себе си Dictionary с всички активни игри и отговаря за създаването, премахването и останалата игрова логика.

За multiplayer частта използвах SignalR за real-time комуникация между сървъра и клиента. Разбира се можех да правя заявки през малък интервал от клиента към сървъра, но това щеше да е голям workaround. Настроил съм SignalR да работи в асинхронна среда, като съм използвал конкурентно достъпване на колекцията, пазеща ID-тата на логнатите играчи, така че не би трябвало да има неочаквани Exception-и, и при по голямо натоварване на сървъра. (Настройката е в „Hangman\Source\Web\Hangman.Web\Hubs\Notifier.cs“).

1. Имплементация на играта на клиента.

Използвал съм готова тема bootstrap. Всички манипулации с DOМ правя посредством jQuery. На клиета се изпраща обект съдържащ информация за текущата игра, като на базата на този обект се показва съответната картинка на бесилката, изписва се думата и се показва информация за останалите играчи.

1. Статистика.

Пази се статистика за всяка приключила игра. Ако играчът напусне играта преждевременно няма да му се запише статистиката. Статистиката се кешира. Сложил съм 1 минута време за кеширане за да се провери, че работи.

1. Crossbrowser support

Тествал съм на Chrome и Firefox. В IE имаше малък проблем със SignalR.

1. Gameplay.

Мисля, че е излишно да описвам в детайли. Логнатите потербители виждат бутоните “New Game” и „Join Game“.

Oт „New Game“ се създава single или multiplayer игра. От „Join Game“ играчът се включва в съществуваща игра. При затваряне или редиректване от прозореца на играта, ако всички играчи затовтят прозореца играта се заличава от активните игри.

1. Seed

За да се тества по-лесно съм вкарал малко начални данни.

Потербители: [user1@site.com](mailto:user1@site.com), [user2@site.com](mailto:user2@site.com), и т.н. до 10. Паролите им са „123456“.

Сложил съм само 2 категории Animals и Towns. Думите са съответно:

Animals: Elephant, Tiger.

Towns: Stara Zagora, Berlin.

Думите се проверяват Case Insensitive.