Obecnie istnieje mnóstwo różnorodnych technologii i narzędzi, które umożliwiają dogodny sposób rozwoju aplikacji internetowych. Dokonanie wyboru pomiędzy nimi jest zazwyczaj uzależnione od relacji koniecznych nakładów do zysku, jakie może wygenerować takie rozwiązanie. Z powodu rosnącej popularności treści zamieszczonych w Internecie, autorzy oprogramowania muszą poświęcić dużo czasu na opracowanie interaktywnego i estetycznego systemu.

Do wykonania opisywanego projektu wykorzystano technologie firmy Microsoft. Technologie te jeszcze do niedawna były wykorzystywane głównie do rozwoju dużych aplikacji biznesowych. Obecnie jednak wraz z obniżeniem cen rozwiązań serwerowych jak i darmowemu udostępnieniu środowiska pracy Visual Studio przez Microsoft następuje wzrost popularności tej technologii.

W celu szczegółówego opisu warstwy prezentacji przedstawiono technologie związane z językiem JavaScript. Umożliwiły one stworzenie estetycznej i interaktywnej warstwy wizualnej w stosunkowo krótkim czasie, spełniając jednocześnie kryteria dotyczące bezawaryjności i skalowalności.

**3.5. AngularJS**

AngularJS jest biblioteką programistyczną, rozpowszechnianą na licencji MIT przez firmę Google. Jest to biblioteka języka JavaScript, przeznaczona do rozwoju aplikacji internetowych. Dziękie zastosowaniu opisywanego narzędzia możliwe jest oddzielenie zmiany struktury dokumenty (DOM) od logiki aplikacji. Poprawne wykorzystanie AngularJS umożliwia oddzielenie warstwy klienckiej od warstwy serwerowej. Od strony architektury, AngularJS implementuje wzorzec MVC (Model-View-Controller) i zaleca utrzymywanie słabych zależności pomiędzy warstwami logiki, prezentacji i danych. Dzięki czytelności kodu w porównaniu do jQuery uzyskano warstwę łatwą do testowania, skalowalną i odporną na awarie.

3.1. Serwer

Przyjęto, że z system będzie rozwijany lokalnie, na komputerach obu autorów. Aby umożliwić uruchomienie aplikacji konieczne było uruchomienie instancji serwera. Posłużono się do tego programem IIS Express wbudowanym w Visual Studio. Jest to zbiór usług internetowych, który podczas prac programistycznych pełnił rolę serwera HTTP i SMTP. Z racji korzystania z platformy .NET niemożliwe było wykorzystanie innego rozwiązania. Można jednak stwierdzić, że IIS Express spełnił swoje zadanie i w żaden sposób nie spowalniał ani nie utrudniał prac programistyczych.

3.2. Platforma .NET

Do wykonania aplikacji skorzystano z platformy .NET. Jest to bogata w funkcje platforma programistyczna, opracowana przez firmę Microsoft. Obejmuje ona środowisko uruchomieniowe oraz biblioteki klas, które są odpowiedzialne za podstawowe funkcjonalności aplikacji. Platforma ta umożliwia programowanie w wielu językach – na potrzeby realizacji projektu przyjęto język C#. Platforma .NET zarządza takimi częściami systemu, jak: kod źródłowy aplikacji, sposób zarządzania pamięcią, obsługa wyjątków oraz zabezpieczenia. Możlwe jest wykorzystanie wyżej opisanej platformy zarówno do aplikacji internetowych, jak i desktopowych. W opisywanym przypadku wybrano środowisko sieciowe, które jest zarządzane przez serwer internetowy IIS (więcej w rozdziale….). W skład platformy .NET wchodzą zarówno kompilatory języków wysokiego poziomu (C#, VB, C++), jak i kompilatory JIT. Nie są one jednak tematem poniższej pracy i nie będą dokładnie opisywane. Kompilator języka C# odpowiada za konwersję do języka pośredniego CIL (Common…) który następnie jest tłumaczony przez kompilator JIT do kodu maszynowego.

3.3. Język C#

Do wykonania aplikacji zastosowano język C#. Jest to silnie typowany, obiektowy język programowania rozwijany przez firmę Microsoft od 2000 roku. Jest jednym z języków, zaprojektowanych na CLR. WHAAAT? Umożliwia on budowę aplikacji z zachowaniem wydzielenia warstw abstrakcyjnych, co jest zgodne z paradygmatem programowania obiektowego. Dzięki zastosowaniu wzorców projektowcyh (patrz punkt 3.9.) możliwa była implementacja funkcjonalności zgodnie z zasadami SOLID. Wzorce umożliwiły również eliminację twardych zależności (patrz rozdział ….).

Na etapie projektu założono, że aplikacja będzie realizowana w metodologii code-first. Polega ona na implementacji klas typu POCO języka C# i konwersji ich na tabele w bazie danych. W celu przeprowadzenia tej konwersji zastosowano Enity Framework, który jest narzędziem typu ORM (Object Relational Mapping). Jest to najpopularniejszy mapper dla platformy .NET dostępny na rynku. Alternatywą była biblioteka Nhibernate, która umożliwia bardzo podobny zbiór funkcjonalności, co EF. Wybór w tym przypadku był podyktowany świetnym wsparciem technologii EF, w porównaniu do Nhibernate oraz wcześniejszym doświadczeniem tej technologii

3.4. LINQ

System ServiceCMS w dużej mierze opiera się na zwinnym zarządzaniu operacjami CRUD (ang..) na zbiorach encji biznesowych. Z tego powodu konieczne było wypracowanie procedury postępowania z operacjami na kolekcjach. Należało mieć na uwadzę zarówno czystość, prostotę kodu jak i kwestie wydajności oraz bezawaryjności. Naturalnym wyborem dla platformy .NET jest rozszerzenie LINQ (ang….). Dostarcza ono bogaty zbiór metod do przeprowadzenia wszelkich operacji na zbiorach obiektów. Dzięki mechanizmowi Lazy Loading (ang. Opóźnione wywołanie) rozwiązanie to spełnia również kryteria wydajnościowe. LINQ zostało uznane za rozwiązanie bezkonkurencyjne i nie rozważano innych rozszerzeń, które mogłyby zostać użyte podczas implementacji ServiceCMS.

3.5. Kontrola wersji

Koncepcja systemu zakładała realizację w dwuosobowym zespole programistycznym. Z tego powodu konieczna była organizacja projektu za pomocą systemu kontroli wersji. Do przechowywania plików źródłowych wykorzystano serwis Github. Jest to najbardziej popularna platforma, umożliwiająca wersjonowanie za pomocą Git-a. Do obługi repozytorium wykorzystano bezpłatny klient Source Tree. Umożliwia on zarówno zatwierdzanie zmian, jak i ewentualny powrót do wcześniejszych faz projektu w przypadku napotkania błędów. Rozważano również wykorzystanie innych klientów przeznaczonych do wersjonowania. Wśród nich były między innymi – TFS, …,…,. Z uwagi jednak na podobieństwo tych rozwiązań wybrano darmowe i nieskomplikowane narzędzie, które jednocześnie realizowało wszystkie założenia projektowe.

3.6. Producteev

W trakcie tworzenia aplikacji konieczny był szczegółowy podział pracy pomiędzy osoby, realizujące projekt. Na etapie projektu wyszczególniono obszary, za które odpowiedzialni będą poszczególni autorzy. Formalna organizacja została przeprowadzona za pomocą aplikacji Producteev. Służy ona do prrzydzielania zadań osobm, biorącym udział w projekcie. Zadanie te posiadają statusy (….) oraz możliwe pod-zadania (ang. Subtask). Te podstawowe funkcjonalności były w zupełności wystarczające do organizacji zadań programistycznych – każda bardziej rozbudowana aplikacja kreowała by jedynie dodatkowe obowiązki, związane z jej obłusgą. Atutem Producteev była również licencja Freeware(?), dzięki której nie generowano kosztów, związanych z projektem.

RestSharp jest biblioteką przeznaczoną dla platformy .NET, która ułatwia przeprowadzanie żądań o architekturze REST do zewnętrznych usług. Dzięki wykorzystaniu tej biblioteki proces przeprowadzenia żądań http oraz deserializacji otrzymanych wyników jest znacznie prostszy. W aplikacji ServiceCMS konieczne było przechwycenie wyników w formacie JSON, co było możliwe dzięki wbudowanemu w RestSharpa serializatorowi JSON. Jest to biblioteka posiadająca bogatą dokumentację i wsparcie online. Powyższe cechy jak i mnogość funkcji z jednoczesnym zachowaniem prostoty obsługi zadecydowały o wybraniu tego narzędzia.

3.7. Wzorce

Unit of work (ang…) jest wzorcem, który ułatwia korzystanie z repozytorium. Dzięki implementacji tego podejścia możliwy jest dostęp do wszystkich repozytoriów z poziomu jednej klasy. Dodatkowym atutem jest fakt, że dzięki wzorcowi Unit of work łatwo zaimplementować obsługę transakcji bazodanowych. Z racji konieczności wielokrotnego przeprowadzania operacji CRUD w bazie danych było to spore ułatwienie i zabezpieczenie przed niespójnością danych.

Wstrzykiwanie zależności jest wzorcem architektonicznym, którego założenia polegają na usuwaniu bezpośrednich zależności pomiędzy komponentami na rzecz wykorzystania powiązań miękkich czyli połączeń za pomocą interfejsów a nie konkretnych implementacji. Z technicznego punktu widzenia wstrzykiwanie zależności jest przpadkiem szczególnym realizacji paradygmatu IoC który polega na odwróceniu odpowiedzialności za kontrolę wybranych operacji. Z praktycznego punktu widzenia zastosowanie tego wzorca umożliwiło ograniczenie koniecznej refaktoryzacji kodu po wprowadzeniu poprawek. Dzięki ograniczeniu twardych zależności nie była konieczna zmiana implementacji logiki po zmianie klasy zależnej. W takim wypadku wystarczyło wstrzyknięcie innej implementacji.

DI silnie preferuje zewnętrzne tworzenie połączeń (zależności) pomiędzy komponentami oprogramowania (np. asocjacje w klasach), nad zlecaniem tworzenia zależności im samym. Jest to wzorzec, w którym odpowiedzialność za tworzenie obiektów i łączenie jest przeniesiona z obiektów do fabryki (np. kontenera IoC).

Repozytorium jest warstwą aplikacji, która izoluje obiektową warstwę dostępu do danych. Działa ono na poziomie jednej klasy modelu (przypis?). Służy ono do uniezależnienia warstwy bazodanowej, która umożliwia dostęp do danych, od implementacji logiki biznesowej. W poniższej aplikacji wykorzystano repozytorium generyczne, które znacznie zmniejszło nakład pracy, potrzebny do implementacji dostępu do danych.

Wzorzec MVC (ang….) jest wzorcem architektonicznym, służącym do organizacji struktury systemów informatycznych. Został on zaprojektowany w 1979 przez programistów, pracujących nad językiem Smalltalk. Zakłada on podział aplikacji na trzy warstwy:

-warstwa Modelu (reprezentacja logiki biznesowej aplikacji).

-warstwa widoku (opisuje sposób prezentacji systemu użytkownikowi)

-warstwa kontrolera (przyjmuje dane z warstwy prezentacji i wywołuje metody zaimplementowane w modułach)

Warstwy te są ściśle zależne pomiędzy sobą. W trakcie implementacji funkcjonalności trzymano się zasad SOLID, co spowodowało marginalizację roli kontrolerów, na rzecz izolacji logii biznesowej w osobnych modułach. Wzorzec MVC zastosowano w celu organizacji procesu implementacji i spełnił on całkowicie oczekiwania i założenia, które sformułowano w fazie projektu.

3.8. VS

Do wykonania projektu konieczne było korzystanie z środowiska programistycznego. Oczywistym wyborem dla języka C# i platformy .NET jest zintegrowane środowisko programistyczne Visual Studio. W poniższym projkecie korzystano z wersji Community, wydanej w 2013 roku. Wybór wersji był podyktowany postanowieniami licencyjnymi firmy Microsoft. Visual studio w znacznym stopniu przyczyniło się do ułatwienia pracy z kodem. Posiada on wbudowany debugger, który był wielokrotnie wykorzystywany do śledzenia występujących błędów. Usługa Nuget umożliwiała bezproblemowe instalowanie i korzystanie z dodatkowych bibliotek (RESTSharp, Ninject). Z racji charakterystyki wykonywanego projektu i systemu operacyjnego, na którym pracowano, niemożliwe było wykorzystanie jakiegokolwiek innego rozwiązania. Można jednak stwierdzić, że jest to narzędzie całkowicie spełniające swoje zadania i umożliwiające ekspresowy rozwój oprogramowania.

3.4. Autofac

Autofac jest kontenerem wstrzykiwania zależności (ang…), opartym na licencji open-source. W realizowanym projekcie został wykorzystany do implementacji wzorca projektowego IoC. Biblioteka ta jest nastawiona na ograniczenie zbędnych funkcji, dzięki czemu jej obsługa nie sprawia problemów. Dzięki bogatej i wyczerpującej dokumentacji próg wejścia i nauki tego narzędzia był relatywnie niski. Rozważano również użycie biblioteki Ninject, która w procesie implementacji nie różniła by się znacznie od wykorzystanej. Na korzyść Autofac-a przemawiała jednak dużo bogatsza dokumentacja i zbiór gotowych rozwiązań, dostępnych w sieci.

(Autofac realizuje procesy zarządzania czasem życia obiektów i rozwiązywania zależności pod maską.)