W obecnych czasach istnieje wiele narzędzi i technologii, które w bezproblemowy sposób umożliwiają rozwój aplikacji internetowych. Wybór jest najczęściej podyktowany ceną i dostępnością danego rozwiązania. Zdarzają się również przypadki, w których specyfika systemu (wydajność, dostępność) dyktuje wybór, który należy podjąć.

Do realizacji przedstawionej aplikacji użyto technologie firmy Microsoft. Narzędzia te były jeszcze stosunkowo niedawno wykorzystywane głównie do rozwoju dużych aplikacji biznesowych. Aktualnie ich popularność rośnie również w odniesieniu do małych i średnich projektów, co jest bezpośrednią pochodną obniżenia cen rozwiązań serwerowych jak i bezpłatnemu środowisku pracy Visual Studio.

W celu przedstawienia części logiki biznesowej zaprezentowano technologie związane z językiem C# i platformą .NET. Umożliwiły one implementacje zaprojektowanych funkcjonalności w sposób bezproblemowy i elastyczny, spełniając jednocześnie kryteria dotyczące bezpieczeństwa i wydajności.

3.1. Serwer

Założono, że aplikacja będzie rozwijana w środowisku lokalnym. Aby spełnić to wymaganie należało uruchomić serwer (?). W związku z faktem, że system był rozwijany w środowisku programistycznym Visual Studio, naturalnym wyborem był program IIS Express. Zawiera on bogaty zbiór usług internetowych. W projekcie wykorzystano powyższy program jedynie do roli serwera HTTP i SMTP. Fakt wykorzystywania platformy .NET uniemożliwiał wykorzystanie innego rozwiązania.

3.2. Platforma .NET

Do implementacji systemu wykorzystano platformę .NET. Jest to technologia rozwijana przez firmę Microsoft. W jej skład wchodzi środowisko uruchomieniowe (CLR?) oraz bogaty zbiór bibliotek klas, które umożliwiają realizację aplikacji. (?) Platforma .NET umożliwia programowanie w wielu językach (m.in. C#, VB, C++). Powyższy projekt zrealizowano w języku C# (opis w pkt..). Platforma .NET odpowiada za kod źródłowy i jego kompilację, oraz sposób alokacji pamięci.

Platformę .NET można wykorzystać do rozwoju aplikacji internetowych, desktopowych(?) oraz systemów wbudowanych. Opisywany system jest systemem opartym o środowisko sieciowe. W skład opisywanej platformy zalicza się kompilatory języków wysokiego poziomu (C#, VB, C++) oraz kompilatory JIT i AoT (CIL do kodu maszynowego). Kompilatory platformy .NET nie są tematem tej pracy i z tego względu nie będą dokładnie opisane.

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 języja 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.

3.4. LINQ

Aplikacja ServiceCMS zależy w dużym stopniu od bazodanowych operacji CRUD (ang…). Z racji tego, niezbędne było opracowanie procedury postępowania z kolekcjami danych. Zwrócono uwagę szczególnie na kwestie wydajności oraz bezawaryjności. Oczywistym wyborem w tym przypadku jest rozszrzenie LINQ (ang…). Jest to biblioteka zawarta w platformie .NET, która zawiera zbiór metod i rozszerzeń klas, przeznaczonych do operacji na kolekcjach. Dzięki wbudowanym mechanizmom optymalizacyjnym (min Lazy Loading i….!!), spełnia ona założone normy wydajnościowe. Biblioteka LINQ jest w opisywanym przypadku rozwiązaniem bezkonkurencyjnym, przez co nie analizowano innych możliwych rozwiązań.

3.5. Kontrola wersji

Na etapie projektu podjęto decyzję o wykonaniu aplikacji w dwuosobowym zespole programistycznym. Z uwagi na to, niezbędna była organizacja pracy za pomocą systemu kontroli wersji (ang….). Do przechowywania plików źródłowych wykorzystano serwis Github. Jest to najpopularniejsza platforma do obsługi wersji projektów informatycznych. Jako klienta wykorzystano aplikację SourceTree. Umożliwia ona zatwierdzanie zmian (ang…) oraz powrót do wcześniejszych wersji (ang. Rollback). Przy doborze narzędzia rozważano wykorzystanie innych klientów wersjonowania plików. Były to między innymi TFS…….Z uwagi na brak istotnych różnic pomiędzy tymi rozwiązaniami, wybrano darmowe i proste narzędzie, które spełniło wszystkie założenia, przyjęte na etapie projektu.

3.6. Producteev

Podczas procesu implementacji niezbędny był podział pracy pomiędzy obu autorów. Przed fazą wdrożenia funkcjonalności, szczegółowo zaplanowano za które moduły będą odpowiedzialni poszczególni autorzy. Formalny podział został wykonany przy pomocy programu o nazwie Producteev. Służy on do przydzielania zadań osobom, które biorą udział w projekcie. Zadania posiadają statusy, które można zmieniać w zależności od procentowej wartości wykonania zadania. Możliwe jest również dodanie zadań podrzędnych (ang. Substask), które ułatwiają organizację przepływu informacji. Powyższe funkcjonalności uznano za wystarczająco do organizacji dwuosobowego zespołu programistycznego. Atutem Producteev jest również darmowa licencja, dzięki czemu nie wygenerowano dodatkowych kosztów.

3.9 RestSharp

RestSharp to darmowa biblioteka dla platformy .NET. Umożliwia ona wykonanie żądań REST do zewnętrznych serwerów w łatwy i elastyczny sposób. Zastosowanie tej biblioteki umożliwiło wykonanie żądań HTTP oraz deserializacje uzyskanych wyników bez konieczności samodzielnej implementacji. W opisywanym systemie jednym z wymagań było przechwycenie odpowiedzi w formacie JSON. RestSharp umożliwia realizację tego wymagania, poprzez implementację serializatora JSON. Dzięki bogatej i przejrzystej dokumentacji korzystanie z tej biblioteki nie powodowało jakichkolwiek problemów. Nie rozważano użycia innego rozwiązania – moduł RestSharp jest najpopularniejszym tego typu dodatkiem dla platformy .NET.

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 Unit of work bezobsługowo (?!) zrządza transakcjami bazodanowymi. Z racji konieczności wielokrotnego przeprowadzania operacji CRUD w bazie danych było to spore ułatwienie.

Wstrzykiwanie zależności jest wzorcem architektonicznym, którego założenia polegają na usuwaniu bezpośrednich zależności pomiędzy komponentami na rzecz wykorzystania powiązań miękkich. Z technicznego punktu widzenia wstrzykiwanie zależności jest przpadkiem szczególnym realizacji paradygmatu IoC (przypatrz się temu zdaniu!). Z praktycznego punktu widzenia zastosowanie tego wzorca umożliwiło ograniczenia koniecznej refaktoryzacji kodu po wprowadzeniu poprawek. Dzięki ograniczeniu twardych zależności nie była konieczna zmiana sposobu wywoływania i definicji sygnatury funkcji.(przypatrz się temu zdaniu!)

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 rolę 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 realizacji systemu wymagany było środowisko programistyczne. Naturalnym wyborem dla języka C# i platformy .NET jest Visual Studio. W opisywanym projekcie korzystano z wersji Community, wydanej w 2013 roku. Wybór wersji był uzależniony od postanowień licencyjnych firmy Microsoft. Środowisko programistyczne Visual Studio wraz z dodatkiem Resharper znacznie przyczyniło się do ułatwienia pracy z kodem źródłowym aplikacji. Wielokrotnie korzystano z wbudowanego debuggera, który śledził występujące błędy. Usługa Nuget umożliwiła korzystanie z zewnętrznych bibliotek (RESTSharp,Autofac). Z uwagi na wykorzystywany system operacyjny i charakter projektu niemożliwe było skorzystanie z innego rozwiązania.

3.4. Autofac

Autofac to kontener wstrzykiwania zależności (ang….). Jest on udostępniany zgodnie z licencją open-source. W opisywanym systemie został wykorzystany do implementacji wzorca projektowego IoC. Autofac jest z założenia biblioteką pozbawioną zbędnych funkcjonalności. Rezultatem takiego podejścia jest stosunkowo prosty schemat implementacji wstrzykiwania zależności. Warto zauważyć, że Autofac realizuje procesy zarządzania czasem życia obiektów i rozwiązywania zależności bez udziału programisty. Dodatkowym atutem tej biblioteki jest bogata dokumentacja, która wyczerpująco opisuje wszelkie problemy, które można napotkać podczas korzystania z niej.