**Specyfikacja Implementacji eAd v.1.0**

Spis Tresci

Inhaltsverzeichnis

[1.Architektura systemu 3](#_Toc490944628)

[2.BE 6](#_Toc490944629)

[2.1 Zalozenia, standardy, wykorzystane technologie i metodologie 6](#_Toc490944630)

[2.2 Zarzadzanie sesja 6](#_Toc490944631)

[3.FE 6](#_Toc490944632)

[4. Baza danych. 6](#_Toc490944633)

# 1.Architektura systemu

1.1 Ogolny zarys systemu

System eAd (Eteczka) jest aplikacja typu WebApplication. Modelem architektonicznym jest MVC (Model – View – Controller). Oznacza to, ze Model (Baza danych, XML, eteczka.main) posiada pewien stan. Stan ten ulega zmianie tylko poprzez dzialania kontrolera (BE – C#). Kontroler odczytuje i zapisuje stan, sam w sobie nie posiadajac zadnej pamieci (zmiennych globalnych, danych „w pamieci“). View – widok – na rzadanie uzytkownika (klikniecie przycisku, wybranie opcji menu) komunikuje sie z kontrolerem i prosi o aktualny stan danych. Zaleta tego modelu jest to, ze wielu uzytkownikow moze rownoczesnie korzystac z systemu i dostac spojne dane. Wada – troszeczke trudniejsza implementacja.

|  |
| --- |
|  |
| Rys.1 Schemat ogolny systemu |

Nasz system jest aplikacja typu klient-serwer. Serwerem calego systemu jest komputer z wgranym systemem operacyjnym Windows lub Windows Server z zainstalowanym kontenerem IIS **(**[**https://pl.wikipedia.org/wiki/Internet\_Information\_Services**](https://pl.wikipedia.org/wiki/Internet_Information_Services)**)**. IIS jest serwerem dla eAd, windows serwer dla calej naszej platformy (eAd + Fox). Gdy uruchamiamy eAd w Visual Studio, uruchamiamy tak naprawde mini-wersje IIS – IIS Express, mniejsza wersje serwera do developmentu (Rys.2)

|  |
| --- |
| d:\Usr\Tokarz\private\eteczka.rel\eteczka\Design\Obrazy\iis express.png |
| Rys. Nr 2 – IIS Express |

Na rys. 1 widac glowne komponenty skladowe systemu oraz sposob i protokol (jezyk) komunikacji miedzy komponentami. Czesc eteczki dzialajacej na serwerze IIS konfigurujemy za pomoca glownego pliku konfiguracyjnego Web.config. Znajduje sie on w projekcie Eteczka

|  |
| --- |
|  |

Plik ten jest konfiguracja naszej aplikacji. Zawiera definicje konfiguracyjne serwera i ustawienia polaczen BE-FE. Dodatkowo zawiera parametry systemu, ktore ustawiamy sami, takie jak connection string czy nazwa zmiennej srodowiskowej do katalogu roboczego eteczki.

Baza danych wybrana do systemu to PostgreSQL i serwer bazy danych zarzadzany przez aplikacje pgAdmin. Serwer ten dziala w tle (jako proces)

|  |
| --- |
|  |

Domyslnie jest dostepny na hoscie lokalnym pod numerem portu 5432 (localhost:5432). Uruchamia sie przy starcie systemu. Do zdefiniowania przez nas pozostaja dane dostepowe u klienta.

1.2 Schemat BE

|  |
| --- |
|  |
| Rys. Nr X Schemat Back End Eteczki |

BE sklada sie obecnie z 2 glownych projektow : BE (Backend) i DB (Database). DB jest odpowiedzialny za warstwe polaczenia z baza danych. BE korzysta z DB za pomoca obiektow DAO (Data-Access Object). Idea takiego obiektu jest udostepnienie klasy, ktora w jezyku obiektowym udostepni jakis zasob bazy danych. Przyklad:

Klasa KatLoginDAO, metoda

*public KatLoginy WczytajPracownikaPoNazwieIHasle(string username, string password)*

Zwraca nam ona obiekt z bazy danych dla danego username i password. Nie interesuje nas na tym etapie, jaka to jest baza danych i jak ma wygladac do niej kwerenda. Mozemy obiektowo wywolac logike dostepu do bazy danych. Formatem wymiany danych miedzy BE a DB jest klasa typu DTO (Data-Transfer Object). Jest to obiekt reprezentujacy wynik kwerendy. Czasami jest on widokiem jednej tabeli czasami kilku (gdy Select uzywa operacji Join i laczy kilka tabel). To samo DTO kierowane jest do FE jako dane dostepne dla uzytkownika.

Glowna warstwa komunikacyjna miedzy FE a BE sa klasy typu Kontroler. Sa to zwykle klasy, zakonczone slowem Controller, ktore dziedzicza po klasie bazowej *System.Web.Mvc.Controller* .

Posiadaja metody zwracajace typ *ActionResult.* Z naszego punktu widzeniajest to nic innego jak JSON.

Kontrolery nie powinny posiadac zbyt wiele logiki, ich zadaniem jest odebranie rzadania z FE i oddelegowanie zadan do „Serwisow“. Serwisy zwracaja wartosc, ktora nastepnie kontrolery przekazuja spowrotem do FE w formacie JSON.

Serwisy to glowne miejsce implementacji logiki. Korzystaja one z klas pomocniczych, takich jak Utils, Model, Mappers, Factories, itd. Ich zadaniem jest wywolanie odpowiednich klas pomocniczych w celu pobrania i obrobki danych dla Kontrolera. Lacza sie one rowniez z DAO zeby uzyskac dane z bazy danych.

DAO sluza do pobierania danych z bazy. Korzystaja one z interfejsu *IDbConnectionFactory*

Ktory za pomoca interfejsu IConnection tworzy connection string do naszej bazy i zwraca obiekt IdbConnection ktory moze wykonywac zapytania do bazy danych. W wersji 1.2 bedzie on rowniez wspieral transakcje. Klasy w przestrzeni nazw „Entities“ sa to klasy odpowiadajace 1:1 tabelom w bazie. Ich nazwy pol i typy powinny pokrywac sie z definicjami tabel w SQL.

1.3 Schemat FE

|  |
| --- |
|  |

Glownym interfejsem pomiedzy FE a BE jest serwis httpService. Posiada on definicje metod GET i POST do komunikacji z BE. Powinien on stanowic jedyne polaczenie pomiedzy kontrolerami Angulara a BE. Ujednolici to przede wszystkim sposob komunikacji i ulatwi testowanie. Serwis httpService korzysta z serwisu sessionService do zarzadzania sesja uzytkownika. SessionService tworzy i przechowuje sesja na caly czas pracy z eAd oraz zamyka sesje na koniec pracy. Sesja zostaje stworzona w momencie w ktorym uzytkownik zaloguje sie do serwisu i zostaje zamknieta, kiedy nastapi wylogowanie badz zamkniecie przegladarki.

Jedynymi obiektami odnoszacymi sie do struktury DOM bezposrednio, powinny byc dyrektywy, wewnatrz metody link. Kontrolery i serwisy nie powinny odnosic sie do obiektow DOM besposrednio.

# 2.BE

## 2.1 Zalozenia, standardy, wykorzystane technologie i metodologie

BE powstaje w mysl zalozen MVC. Do tworzenia obiektow wykorzystujemy kontener DI (klasa UnityConfig.cs). Kazda klasa powinna zostac przetestowana. Poczatkowym punktem systemu jest klasa Global.asax w projekcie Eteczka. Startuje ona aplikacje oraz konfiguruje wszystkie parametry systemu. Ze wzgledu bezpieczenstwa, zeskanowane pliki znajduja sie w folderze roboczym wskazywanym przez zmienna srodowiskowa EAD\_DIR. Domyslnie powinien byc to katalog C:eteczka.main

## 2.2 Zarzadzanie sesja

Sesja powstaje podczas zalogowania sie do systemu. Odpowiedzalna za to jest klasa *SesjaController.* Sesja powinna miec timeout i po czasie dluzszej bezczynnosci powinno nastapic automatyczne wylogowanie i zamkniecie sesji. Obiektem kontrolujacym sesje jest klasa *StanSesji.*

2.3 Polaczenie do bazy danych

Polaczenie do bazy danych wywolujemy za pomoca sterownika ODBC. Jest to metoda wykorzystujaca connection string do nawiazania polaczenia. Dane bazy zdefiniowane sa w pliku Web.config a connection string powstaje w klasie *Connection* po wywolaniu metody

*public string GetConnectionString(IConnectionDetails connectionDetails)*

2.3 Komunikacja z FE (Kontrolery)

2.4 Klasy pomocnicze (Serwisy, Mapery, Wrapery, Fabryki)

2.5 Eteczka.main

# 3.FE

# 4. Baza danych.

5. Deployment (Uruchomienie programu na serwerze)

Serwer IIS zlokalizowany jest w systemach Windows Professional i Windows Server na dysku c: pod nazwa inetpub (c:/inetpub). Gdy chcemy uruchomic „deployment“ naszej aplikacji, wybieramy w Visual Studio opcje publish (prawy przycisk myszki na projekcie glownym – u nas to jest projekt Eteczka)

|  |
| --- |
| d:\Usr\Tokarz\private\eteczka.rel\eteczka\Design\Obrazy\publish.png |
| d:\Usr\Tokarz\private\eteczka.rel\eteczka\Design\Obrazy\publish dalej.png |

Gdy proces ten wykona sie bez bledow, w katalogu inetpub powstanie (lub uaktualni sie) katalog ead

|  |
| --- |
| d:\Usr\Tokarz\private\eteczka.rel\eteczka\Design\Obrazy\ead nkatalog.png |

W srodku jego struktura wyglada tak:

|  |
| --- |
|  |

Katalog bin posiada skompilowany BE (pliki dll ). Reszta to pliki FE i ustawienia aplikacji (Web.config). Teraz musimy sie upewnic ze IIS jest uruchomiony:

|  |
| --- |
|  |

Teraz nasza aplikacja dostepna jest na serwerze pod adresem:

localhost/ead/index.html

|  |
| --- |
|  |

6. Specyfikacja BE

7. Specyfikacja FE