

Технически университет – Варна



Факултет по изчислителна техника и автоматизация

ДИПЛОМНА РАБОТА

Тема

Разработване на система за изготвяне на изискванията и
проектирането на софтуерни проекти

Дипломант :.....

/Щилиян Александров Узунов/

Ръководител:.....

/доц. Д-р Владимир Николов/

Съдържание

Глава 1 Въведение	3
1.1 Въведение в дипломното задание.....	3
1.2 Постановка на дипломното задание.....	3
1.3 Цели и задачи на разработката	4
1.4 Необходимост от решаването на дипломната задача	4
Глава 2 Описание на използваните програмни средства и технологии	5
2.1 Кратък преглед на използваните програмни средства и технологии.....	5
2.2 Детайлен преглед на използваните програмни средства и технологии	5
Глава 3 Описание на програмното решение	25
3.1 Описание на клиентската част	25
3.2 Описание на сървърната част	40
3.3 Описание на базата от данни	44
Глава 4 Ръководство на програмиста	49
4.1 Инсталиране на системата	49
4.2 Използване на системата	50
Глава 5 Заключение, предложения за развитие и изплзване	61
Изходен код на системата	62
Използвана литература	135

Глава 1

1.1 Въведение в дипломното задание.

С развитието на интернет стандартите и технологиите приложенията които се изпълняват от web browser-ите стават все по-мощни и с все по-големи възможности. Сравнени с desktop приложенията, web приложенията са достъпни по всяко време, от всяка точка на земята, не се нуждаят от инсталиране или поддръжка, не заемат място на потребителския компютър. Също така web базираните приложения са много платформени ,което позволява на потребители с всякакви операционни системи и устройства да се използват приложението. Разбира се сравнени с desktop приложенията те също така имат и недостатъци. Като основни недостатъци могат да бъдат изтъкнати по-бавното изпълнение на изпълнимия код заради наличието на междинен слой между приложния код и операционната система – browser-a, също така и наличието на постоянна(в някои случаи бърза) интернет връзка. На база на тези факти изборът дали дадено приложение да е web или desktop базирано зависи най-вече от изискванията за бързодействие на програмата. Ако извършваните от приложението сметки ще бъдат тежки, и е изключително важно забавянето да бъде сведено до минимум – то подходящото решение трябва да бъде desktop базирано, в противен случай – могат да бъдат използвани предимствата на web базираните приложения.

С увеличаващите се нужди в световен мащаб от IT специалисти, все повече и повече IT фирми започват да набират чуждестранни програмисти през интернет, и да им задават програмни задания през интернет. Тъй като физически програмистите са разделени, комуникацията между тях е затруднена. Това би могло да доведе до лоши резултати при обсъждането на архитектурата и спецификациите на зададеното им приложение, за това използването на подобно приложение, което да позволява работа на много програмисти върху общи файлове за спецификации и диаграми, в процеса на разработка се оказва полезен инструмент, водещ до намаляване на възможните грешки които биха могли да възникнат по време на имплементацията.

1.2 Постановка на дипломното задание.

Разработеният проект представлява web базирана система за определянето на дизайна и спецификациите на проекти. Тя трябва да дава възможност на програмистите бързо и сигурно да могат да достъпват общите документи и диаграми на проекта, да могат да четат/създават/променят/и изтриват документи и диаграми, да може да се налага контрол от старшите на младшите програмисти по отношение на операциите които те могат да извършват по промяната на проектните файлове.

1.3 Цели и задачи на разработката.

Целта на системата е да спомага за определянето на спецификациите и проектирането на проекти. Тази цел е реализирана като на потребителите(програмисти) се дава възможност да четат/създават/променят/изтриват документи и диаграми. Едно от основните предимства на софтуера е възможността за работата върху едни и същи ресурсни файлове от страна на програмистите, без нужда от синхронизация. Като допълнителни функции са заложили възможностите за раздаване на права от старшите програмисти на младшите програмисти, с цел контрол над съдържанието на файловете в проекта.

1.4 Необходимост от решаването на дипломната задача.

Въпреки че интернет съществуват множество приложения, които дават възможност на потребителя да създава различни видове диаграми и текстови документи, никое от тях не е предназначено конкретно за програмисти, и не комбинира двете възможности в една. Масово за създаване на дизайн на приложения се използват desktop базирани приложения, но проблемите с тях са свързани със синхронизацията на файловете, ако повече от двама програмисти искат да променят дадена UML диаграма, то те трябва да използват система за контрол на версиите над файловете (version control system) което усложнява процеса, или дори да си ги препращат един на друг през мрежата, което е доста не удобно.

Поради факта че системата е web базирана, се спестяват неудобствата на програмистите от инсталиране и деинсталиране на друг софтуер на техните компютри, както и от актуализиране на последни версии на приложението.

Изброените тук предимства на web системата и недостатъците на конкурентните desktop базирани приложения я правят полезен в ръцете на всеки колектив от програмисти.

Глава 2

Описание на използваните програмни средства и технологии

2.1 Кратък преглед на използваните програмни средства и технологии

- Мрежови услуги
- MS SQL Server – сървърна система за работа с база данни
- ASP.NET MVC - .Net базиран сървърен framework за създаване на web приложения
- ASP.NET Web API - .Net базиран сървърен framework за създаване на web приложения
- Entity Framework - .Net базиран framework за осъществяване на връзка и комуникация с база данни
- AngularJS – JavaScript базиран framework за създаване на клиентската част на single page web приложения
- Bootstrap – CSS framework за стилизиране на клиентската част на web приложения

2.2 Детайлен преглед на всяка една от използваните технологии

2.2.1 Мрежови услуги

Мрежовата услуга представлява бизнес логика разположена в някой сървър някъде в интернет, която е достъпна през стандартните интернет протоколи – HTTP или SMTP. На практика това са функции „публикували“ интерфейс със своите услуги и методи в мрежовото пространство. Обикновено те са свързани с някаква среда за обслужване. Чрез интернет услуга, може да ес създаде лесно влизане в системи и използване на техните услуги от голямо разстояние. Предимствата на .Net услугите са, че са обектно ориентирани, може лесно да се прокарат през защитната стена на PC-то на компютъра и позволяват по-гъвкава разработка на различните софтуерни системи.

Поведенческите характеристики на web service-ите са следните :

- Слабо съчетани – потребителите на интернет сървиса не са обвързани с него директно. Интерфейса на интернет услугата може да се промени с течение на времето, без да се нарушат възможностите на клиента да си взаимодейства със сървиса. Създаването на свободно свързани архитектури е с цел да се направят софтуерните системи по-лесно управляеми, да се постигне по-лесно интеграция между различните системи, и да може сървърният код да бъде по-лесно преизползван от различни клиенти. Пример за това е използването на едни и същи web service-и от мобилен клиент, browser, или desktop приложение.
- Едрозърнестост – Обектно-ориентираните технологии като C# излагат своите услуги чрез индивидуални методи. Един индивидуален метод е твърде малка операция, за да предостави някаква полезна функционалност на корпоративно ниво. Изграждане на C# система от нулата изисква създаването на няколко малки(фини) методи, които след това да се обединят в едрозърнеста услуга, която се използва или от клиент или от друга услуга. Технологиите на интернет услугите предлагат естествен начин за дефиниране на едрозърнести услуги, които осигуряват необходимото количество бизнеслогика.
- Възможност да бъдат синхронни или асинхронни – Синхронен сървис се отнася до свързването на клиента с изпълнението на услугата. В синхронни извиквания, клиентът се блокира и чака услугата да завърши своето изпълнение преди да продължи. Асинхронни операции позволяват на клиента да извика дадена услуга и след това да продължи да изпълнява други операции. Асинхронните клиенти извличат резултата, който им е необходим, в един по-късен момент, а синхронните клиенти се блокират и чакат за пристигане на резултата. Асинхронността е ключов фактор, за да може да се реализира свободно свързани системи.
- Поддържат отдалечено извикване на процедури (RPC) – Интернет услугите позволяват на клиентите да извикват процедури, функции и методи на отдалечени обекти с помощта на XML-базиран протокол. Отдалечените процедури разкриват входните и изходните параметри, които интернет сървисът, трябва да поддържа.

Интернет сървисите могат да се разделят грубо на 2 групи – SOAP базирани и REST базирани. Разликата не е голяма, защото SOAP базираните сървиси предоставени чрез HTTP са всъщност специален случай на REST-базирани сървиси.

SOAP (Simple Object Access Protocol) е XML(Extensible Markup Language) диалект, който представлява съобщенията и съдържа документите, които се пращат. В SOAP базирани мрежови услуги, SOAP е с невидима инфраструктура. Например, в един

типичен сценарий, наречен заявка/отговор модел на обмен на съобщения (MEP – Message exchange pattern), основната SOAP библиотека на клиента изпраща SOAP съобщението като мрежова заявка, а SOAP библиотеката на интернет услугата изпраща друго SOAP съобщение като отговор заедно с необходимата (изисквана) услуга.

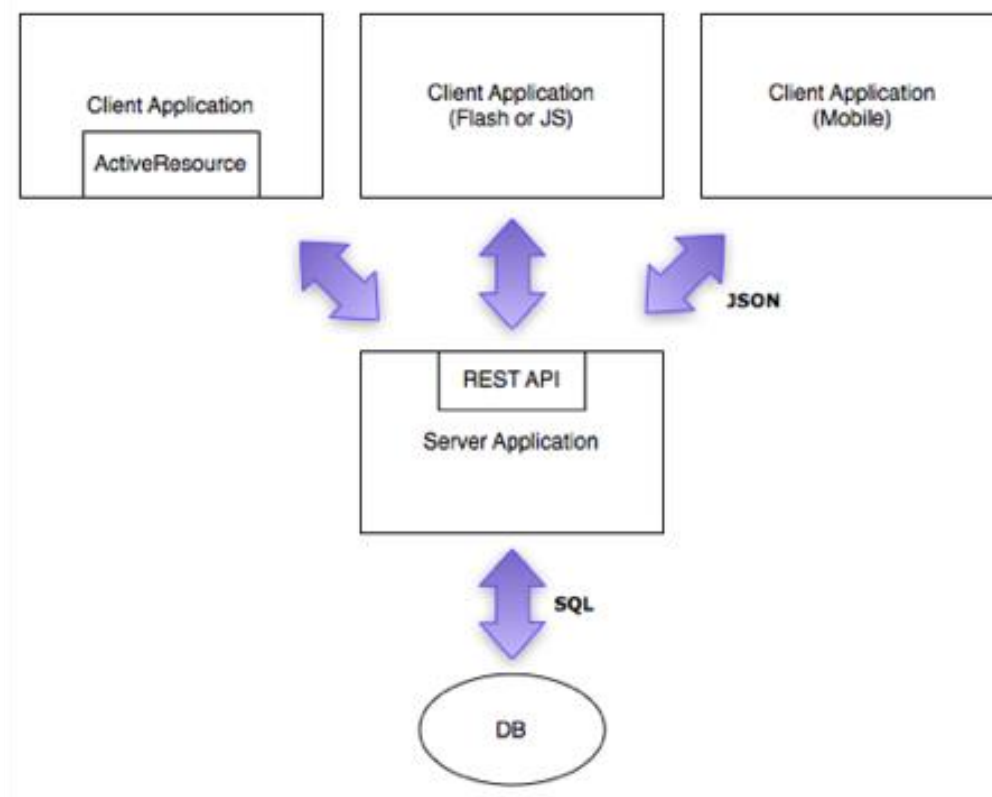
REST (Representational State Transfer) е представен от Рой Фийлдинг като част от неговата дисертация, която анализира различни софтуерни начини за комуникация в разпределени системи през интернет. REST протоколът използва HTTP връзка и методи за комуникация :

- За да се създаде или изпрати ресурс/запис на сървъра се използва POST.
- За да се прочете запис, се използва GET.
- За промяна на стойността на запис се използва PUT.
- За премахване на записа се използва DELETE.

Разлики между SOAP и REST технологиите

- SOAP е XML базиран протокол за предаване на съобщения, REST не е протокол, а архитектурен стил.
- SOAP има дефинирани стандарти, а REST няма.
- SOAP е XML базиран протокол и комуникацията става чрез съобщения, а REST не използва този формат.
- SOAP има силно дефинирани спецификации за всяка част от имплементацията на сървиза. REST има концепция, но няма строги дефинирани правила при имплементацията на сървиза.
- SOAP използва интерфейси и именувани операции, за да изпълни бизнес логика. REST използва (обикновено) URI и методи като (GET, PUT , POST, DELETE) да предаде/изложи ресурси.
- SOAP предоставя инструменти за услугите да се „опишат“ пред „клиентите“ и да съобщят за тяхното съществуване. REST няма такива инструменти и „клиентите“ трябва да знаят за съществуването на услугите.
- Поради това че REST е по-лек като имплементация – той е и по-бърз, докато SOAP е по-бавен, но по-защитен.

След провеждане на необходимото изучаване на двата типа сървиси, в случая с разработването на системата за дизайн и спецификации на софтуерни проекти, подходящата технология е REST.



фиг. 2.2.1.1 REST Архитектурен стил

Допълнителна информация за REST архитектурите.

Архитектурният стил на "REST" прилага шест ограничителни условия, като същевременно дава свобода за дизайна и имплементацията на индивидуалните компоненти:

Клиент-сървър

Единният интерфейс разделя клиента и сървъра. Това означава, например, че клиента не се грижи за складирането на данни. Тази задача остава изцяло за сървъра, като по този начин се подобрява портативността на клиентския код (може да се използва в различни среди). Сървърът няма връзка с потребителския интерфейс и по този начин е по-семпъл и лесен за премащабиране. Клиентът и сървърът могат да бъдат заменени или развивани независимо един от друг, стига това да не налага промяна на единния интерфейс помежду им.

Без статус на сесията (*на английски: Stateless*)

Следващото условие е на сървъра да не се запазват статуси на сесиите. Всяка заявка от клиента, съдържа в себе си нужната информация за нейната обработка, статуси на сесии се запазват единствено при клиента.

Кеширане

Клиента има право да кешира (запазва) информация, получена в отговор от сървъра, за да я преизползва при последващи заявки. За тази цел сървъра трябва имплицитно или експлицитно да е посочил дали информацията в отговора може да се кешира, за да се избегнат случаи, в които клиента получава грешна информация при бъдещи заявки. При правилно управление и използване на кеширането могат частично или напълно да се елиминират ненужни взаимодействия между клиента и сървъра, като по този начин се подобрява бързината и производителността.

Многослойна система

Обикновено клиентът не знае дали е свързан с крайния сървър или със сървър-посредник. Сървърите-посредници подобряват ефективността, като увеличават капацитета за обработване на заявки и предоставят споделени кешове. Също така те допринасят да подобряването на сигурността.

Код при поискване (незадължително)

Сървърът може временно да разшири функционалността, изпращайки код, който се изпълнява директно при клиента. Например клиентски скриптове, написани на JavaScript или компилирани компоненти като Java applets.

Единен интерфейс

Единният интерфейс между клиента и сървъра разделя и опростява архитектурата. По този начин всеки компонент може да се развива самостоятелно.

Единственото условие на REST архитектурата, което не е задължително е "Код по поискване". Всяко приложение (услуга), изпълняващо на гореописаните условия, може да се нарече "RESTful". Ако нарушава дори едно от условията, то не може да бъде считано за "RESTful".

2.2.2 Microsoft SQL Server

MS SQL Server е система за управление на релационна база от данни разработена от Microsoft. Като база данни, това е софтуерен продукт, чиято главна цел е да позволява запазване и извличане на данни по поисква на други софтуерни приложения, без значение дали те ще са на същия компютър или на друг, работещ на друго място в мрежата. Основният език за заявки е T-SQL и ANSI Sql.

Запазване на данните

Начинът на съхранение на данните е чрез колекция от таблици с типизирани колони. SQL Server поддържа различни типове данни, включително основните типове – като например Integer, Float, Decimal, Char, Varchar, Binary, Text и много други. Закръглянето на float-инг числата към целочислени числа става или със аритметично аритметическо закръгляне, или чрез симетрично закръгляне с фиксиране зависещо от аргументите.

MSSql server също позволява и потребителя сам да си дефинира типове за използване. Също биват предоставяни и сървърни статистики като виртуални таблици и изгледи (наречени динамично менежирани изгледи или DMVs). В допълнение към таблиците, базата данни също може да съдържа и други обекти включително изгледи, вложени процедури, индекси и ограничения, също както и регистър с транзакциите. SQL Server база данни може да съдържа максимум 2^{31} обекта, и може да се разположи няколко файла на нивото на операционната система с максимална големина на файла от 2^{60} байта (или 1 ексабайт). Данните в базата данни са съхранени в първични файлове с разширение .mdf. Вторичните даннови файлове, идентифицирани със .ndf разширение, се използват за запазване на опционални мета данни. Регистрационните файлове се идентифицират със .ldf разширение.

Мястото за съхранение алокирано от базата данни е разделено на последователно номерувани страници, всяка с големина 8kb. Страницата е основната единица за вход/изход във операциите на MS Sql Server. Страницата е маркирана със 96байтов хедър който съдържа метаданни за страницата, включително номерът на страницата, типът на страницата, свободното място на страницата и ID на обекта който я притежава. Типът страница дефинира данните съдържащи се в нея – данни зачазени в базата данни, индекс, алокирани карти които държат информация за това колко страници са алокирани в таблиците и индексите, карти с промени които съдържат информация за направените промени по други страници от последния backup или регистриране, или съдържат големи даннови типове като например снимки или текст.

Докато страницата е основната единица за входно/изходна операция, разстоянието е обикновено управлявано по отношение на степента която се състои от 8 страници. Обект в базата данни може да обхваща всички 8 страници, или да споделя степента със до 7 други обекта. Ред в базата данни не може да обхваща повече от 1 страница, така че е ограничен до 8KB по размер. Въпреки това ако данните надминат 8Kb по размер и редът съдържа Varchar или VarBinary данни, данните в тези колони биват преместени към нова страница (или по възможност последователност от страници, наречена алокираща единица) и заместени със указател към данните.

За физическо запазване на таблица, нейните редове се разделят на серия от партиции (номерирани от 1 до n). Големината на партицията е дефинирана от потребителя. По подразбиране всеки ред е една партиция. Таблицата се разделя на няколко партиции с цел да може да се разпредели базата данни върху клъстер. Редовете във всяка партиция се пазят или във B-Дърво или във хийп структура. Ако таблицата има асоциативен индекс за позволяване на по-бърз достъп на редовете, то редовете се запазват подредени според индексните си стойности, с B-дървото предоставящо индекса. Данните са във листовия възел, и другите възли съдържащи индексната стойност за листовата дата достъпни от съответните възли. Ако индексът не е сгрупан, редовете не се сортират според индексните стойности. Индексиран изглед има същата структура на запазване на данните както и индексиранията таблица. Таблица без индекс се запазва в неопределена хийп структура. Както хийпа, така и B-дърветата могат да обхванат няколко алокиращи единици.

Управление на буфера

SQL Server буферира страниците в рамта за да минимизира операциите по писане и четене от диска. Всяка 8kb страница може да бъде буферирана в паметта, множеството от всички страници които в момента са буферирани се нарича буферен кеш. В зависимост от наличната му памет, SQL Server решава колко страници ще бъдат буферирани в паметта. Буферният кеш се менижира от буфер мениджър. Дали ще се чете или ще се пише към страницата се свежда до копирането и в буферния кеш. Последователни четения или писания са пренасочени към копието в рамта. Страницата се обновява на диска от буферния мениджър само ако рам кешът не е бил рефериран от известно време. Докато се пишат страници обратно към диска, се използва асинхронен вход/изход чрез което входно изходната операция се извършва на задна нишка така че да не се налага на други операции да чакат входно изходната операция. Всяка страница се записва заедно със чексума когато е записана. При обратно четене на страницата, чексумата се изчислява отново и се сравнява със запазената версия за уверение че страницата не е била повредена или подправена междувременно.

Конкурентност и заключване

SQL Server позволява на няколко клиента да използват една и съща баз данни конкурентно. Това налага нуждата от контрол на конкурентния достъп към споделените данни, за осигуряване на интегритет на данните - когато няколко клиента обновят едни и същи данните, или клиентите се опитат да четат данни които са във процес на обработка от друг клиент. SQL Server предоставя 2 мода на контрол над конкурентността - песимистична конкуренция и оптимистична конкуренция. Когато песимистичната конкуренция се използва, SQL Server предоставя конкурентния достъп използвайки ключове. Ключовете могат да бъдат споделени или ексклузивни. Ексклузивният ключ предоставя на потребителя ексклузивен достъп до данните – никой друг потребител не може да достъпва данните докато ключът е пуснат. Споделените ключове се използват когато данните се четат – няколко потребителя могат да четат от данните заключени със споделен ключ, но не могат да получат ексклузивен ключ. Последния ще трябва да изчака за всички споделени ключове да бъдат освободени. Ключовете могат да бъдат приложени на различни нива на грануларност – на цели таблици, страници или дори на редова основа. За индекси, може да бъде на целия индекс или на индексните листа. Нивото на грануларност което се използва се дефинира на база на цялата база данни от администратора на базата. Докато по-финно заключване на системата позволява повече потребители да използват таблиците едновременно, то изисква и повече ресурси. SQL Server също включва две по-леки решения – решета и спинлокс – които са по-малко здрави от ключовете но изискват по-малко ресурси. SQL Server ги използва за DMVs и други ресурси които обикновено не са много заети. SQL Server също наблюдава работата на работните нишки които получават ключове за да осигури че те не свършат като deadlock-ове. В такъв случай SQL Server взема изчерпвателни мерки, които могат да доведат до убиването на някоя от тези нишки включени в deadlock-а и изпълняване на ролбек на транзакцията която е започната. За импелментирането на механизма на заключване, SQL Server съдържа Lock Manager. Той поддържа таблица в паметта която управлява обектите в базата и ключовете. Достъпът до някой от споделените обекти се осъществява като се мине през lock менаджера, който или дава достъп до ресурса или го блокира.

SQL Server също предоставя оптимистичния контрол над конкурентността, който е подобен на многоверсионния конкурентен контрол използван в другите бази данни. Механизмът позволява на новите версии на редовете да бъдат създадени която редът е обновен, за разлика от пренаписването на ред, примерно редът е допълнително идентифициран от ID на транзакцията която е създавала версията на реда. И двете – новата и старата версии на реда се пазят и поддържат, въпреки че старите версии се местят извън базата данни във системната база идентифицирана като Tempdb.

Когато един ред се обработва, всякакви други заявки не се блокират но се изпълняват на старата версия на реда. Ако другата заявка е за обновяване, резултатът ще бъдат две новеи версии на редовете – и двете ще бъдат запазени в базата данни, идентифицирани от съответните им ID-та за транзакцията.

2.2.3 ASP.NET MVC

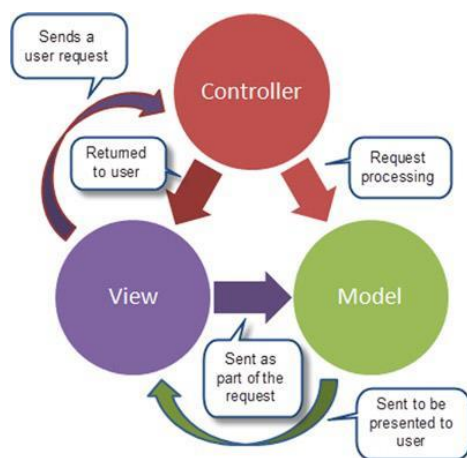
ASP.NET MVC е web application фреймуърк с отворен код, който имплементира MVC шаблонът. Базиран на ASP.NET, ASP.NET MVC позволява на софтуерните разработчици да строят web приложения като композиция от три роли : Модели, Изгледи и Контролери. MVC шаблонът дефинира web приложенията със три логически слоя :

- Модел (Бизнес слой)
- Изглед (Показан слой)
- Контролер (Входен слой)

Моделът представлява състоянието на конкретен аспект от приложението. Задачата на контролерът е да обработва взаимодействията с потребителя и да обновява модела в съответствие със промяната със състоянието на приложението, и след това да предаде информацията на изгледа. Изгледа приема нужната информация от контролера под формата на модел за изгледа и показва на потребителя нужната информация.

През април 2009г. Сорс кода на ASP.NET MVC е пуснах публично под публичния лиценз на Microsoft (Microsoft Public License MS-PL).

ASP.NET MVC е лек, високо надежден и лесен за тестване презентационен фреймуърк, който е интегриран със съществуващите ASP.NET функционалности. Някои от тези интегрирани функционалности са „мастър“ страниците и „членово-базираната“ автентикация. MVC фреймуърка е дефиниран във асембли файла System.Web.Mvc.



Фиг 2.2.3.1 MVC архитектурен стил

Свързването на моделите, изгледите и контролерите е направено чрез интерфейсно базирани „договори“, като по този начин се позволява на всеки компонент да бъде тестван отделно, без нужда от навързване с някой от другите компоненти.

ASP.NET MVC изгледите използват Razor view engine-а за дефиниране на съдържанието, което да се покаже на крайния клиент. Razor view engine-а представлява програмен синтаксис чрез който динамично може към дадената .html страница да се добави съдържание. Използваните езици при razor engine-а са C# или VB.

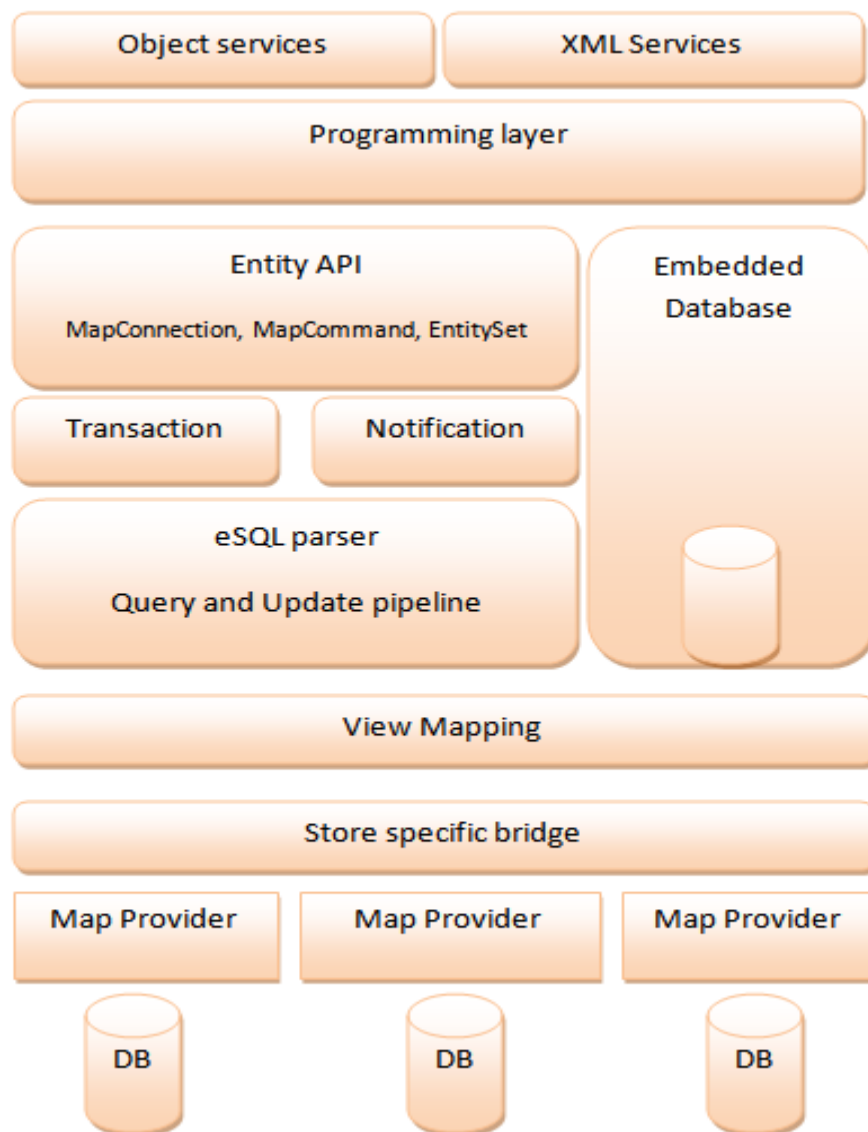
Основното приложение на ASP.NET MVC е за динамично създаване на съдържанието на страници. Именно и това е разликата му със ASP.NET Web API, който е предназначен за създаване на REST базирани Web Service-и.

2.2.4 Entity Framework

Entity Framework (EF) е обектно-релационно съпоставящ фреймуър с отворен код предназначен за ADO.NET, част от .Net фреймуърка. Представлява съвкупност от технологии във ADO.Net които поддържат разработката на данново-ориентирани приложения. Архитектие и разработчиците на данново-ориентирани приложения обикновено имат проблеми със нуждата да постигнат две много различни цели. Те трябва да моделират обектите, връзките и логиката на бизнес проблемите които решават, и трябва също така да работят с бази данни за запазване и четене на данни. Данните могат да се разполагат на няколко системи за запзване, всяка с различни протоколи, дори приложения които работят със единична система за запазване трябва да балансират изискванията на системата за запазване срещу изискванията на писане на ефикасен и лесен за поддържане код на приложението.

Entity Framework позволява на разработчиците да работят с данните под формата на обекти конкретни за езика който ползват, без да се притесняват за стоящата под тях база данни, таблици или колони в които данните се пазят. Entity Framework предоставя на разработчиците едно по-високо ниво на абстракция на което те могат да обработват данните, и да създават данново-ориентирани приложения с по-малко код от колкото ако сами трябва да си пишат връзката с базата данни, или заявките към нея. Тъй като EF е част от .Net фреймуърка, EF приложенията могат да вървят а всеки компютър на който има инсталиран .Net framework.

Архитектура



Архитектурата на ADO.NET Entity Framework-а от долу на горе се състои от следното :

- База данни към която фреймуърка да се свърже. ADO.NET интерфейсите абстрахират връзката с тази база данни когато се програмира според концептуалната схема на EF.
- Map provider – специфичен за базата данни provider който превежда Entity SQL командата на заявка, която е специфична за долу лежащата база данни. Този provider включва Store Specific Bridge, който е компонента отговорен за превеждането на родови команди към специфични команди на запазващата среда.
- EDM парсер и view маппинг, който взема SDL спецификацията на данновият модел и начина по който той се мапва към отдололежащия модел и позволява

програмирането срещу този концептуален модел. От релационната схема, той създава изгледите към данните в съответствие със концептуалния модел. Той агрегира информация от няколко таблици за да може да ги събере в един обект, и иг разделя и обновява обекта в различните таблици към които той принадлежи.

- Линия за заявки и update – обработва заявките, филтрите и подновява заявките за да ги превърне във канонични командни дървета които след това се превръщат във специфични заявки в зависимост от базата данни от мап провайдера.

- Метадата услуги – които управляват всички метаданни свързани с обектите, връзките и съотношенията.

- Транзакции- за интегриране на транзакционните възможности на отдолу лежащата база данни. Ако от базата данни не се поддържат транзакции, то на този слой има точки за разширение с които програмистите могат сами да си ги дефинират.

- Коцептуален API слой – това е изпълнимата средак оято предоставя програмния модел за програмиране срещу концептуалната схема. Той следва ADO.NET шаблона на използване на обекти за конекции за връзка към мап провайдера, използвайки командни обекти за ипращане на заявки и връщайки EntityResultSets или EntitySets съдържащи резултата.

- Разкачени компоненти – които локално кешират множества от данни и множества от обекти.

- Вградена база данни – ADO.Net Entity Framework включва в себе си база данни за клиентско кеширане и правене на заявки към релационни данни.

- Инструменти за дизайн – такива като Mapping Designer, които също са включени във ADO.NET EF служат за олесняване на работата на съпоставяне на концептуалната схема към релационната схема и специфицирането на които свойства на обектния тип трябва да съответстват на коя таблица и ред в базата данни.

- Програмен слой – който предоставя EDM като програмна конструкция, която може да бъде използвана от програмните езици.

- Обектни услуги – автоматично генерират код за CLR класове, които предоставят същите свойства като обекта, и по този начин преудоставящи възможност за инстанциране на ентититата като .Net обекти.

2.2.5 AngularJS

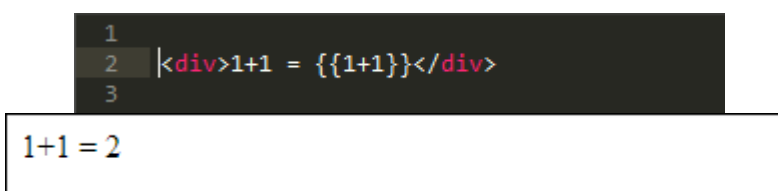
AngularJS е уеб фреймуърк с отворен код, поддържана от google, която спомага в създаването на едно single page приложения, които изискват HTML CSS и JavaScript на клиентската страна. Целта и е да разшири web приложенията със MVC модела, с цел – да направи разработката и тестването по-лесни.

Библиотеката чете HTML който съдържа допълнителни тагове от стандартните тагове описани в HTML спецификацията, след това спазва директивите в тези допълнителни атрибути, и свързва входните и изходните части на страницата към модел, който е представен в стандартни JavaScript променливи. Стойностите на тези Javascript променливи могат да бъдат зададени ръчно, или да бъдат взети от статични или динамично JSON ресурси.

Изрази

AngularJS позволява правилното конструиране на Web Приложения много лесно. За тази цен AngularJS съдържа няколко концепции за да раздели различните части на приложението.

За да се създават лесно изгледните на приложението, AngularJS позволява изразите да бъдат директно вградени във изгледа на HTML страницата. В тези изрази, имаме достъп до JavaScript код, който ни дава възможност да реализираме изчисления за да покажем това от което имаме нужда. Фигурата долу представлява пример за вграден израз, и резултатът от него.



Изразите във HTML страниците са полезно нещо, но създаването на цяло web приложение само с тях би било доста трудно за писане и поддръжка. Изразите се използват за малки операции. За да се структурира приложението добре AngularJS предоставя доста по-удобен инструмент – директивите.

Директиви

Директивите са едно от най-мощните средства на AngularJS. Те Ви позволяват да разширите HTML таговете, в отговор на нуждите на Web приложението. Директивите Ви позволяват да определите как ще бъде структурирана страницата Ви за данните които са налични в определен обхват.

Директивите са нещо което ви предоставя нов синтаксис. Директивите са маркери на DOM елементите, които добавят специално певедение към елемента. Примерно в статичен HTML код нямаме маркер за създаване на календар. За да „научим“ HTML На този нов синтаксис трябва да създадем директива за календар. Директивата по някакъв начин ще създаде елемент, който се държи като календар.

Data binding (Данново свързване)

AngularJS не само че позволява да се структурират изгледите с директиви, но и дава възможност за дафиниране на данново свързване между данните в модела и съдържанието на изгледите. По този начин всяка промяна върху модела – била автоматично прехвърлена върху изгледа, без нужда от експлицитно писане на код от страна на програмиста, който да обхваща този случай, и обратното – при промяна в изгледа, модела бива автоматично променен. Data binding-а е техника която изключително много ускорява писането на приложението, и подобрява качеството на кода, правейки го по-лесно четим, лесен за поддръжка, премахвайки ненужният код за синхронизация между Model и View.

Филтри

Филтрите в AngularJS са функции, които ни позволяват да филтрираме дадена колекция, по подадено правило. Логиката за филтрирането се описва във филтър функцията. Често филтрите се използват във ng-repeat директивата, и по този начин с помощта на данново свързване между модела и изгледа, модела бива филтриран по подадената филтрираща функция, след което резултата който се показва във изгледа съответства със модела.

Обхвати (Scopes)

Обхватът е специфичен модел който се отнася към приложният модел. Той е изпълнимият контекст на изразите. Обхватите са подредени във йерархична структура, която прилича на DOM структурата на приложението. Обхватите наблюдават изрази, и издават събития при промяна на изразите.

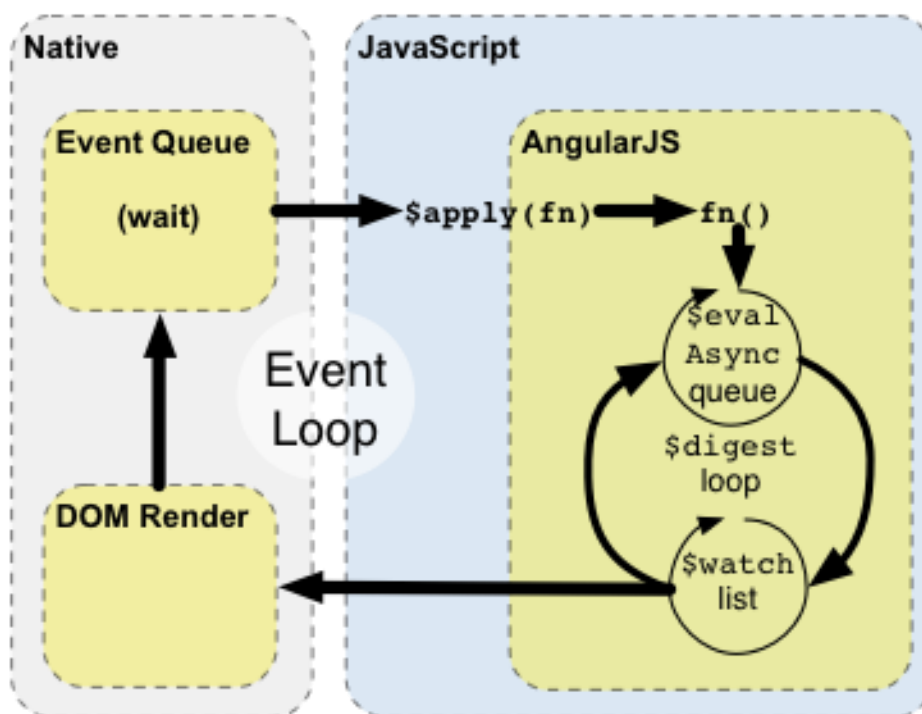
Живот на scope обекта.

Нормалният начин при който браузъра отработва събития е чрез изпълнение на съответстваща колбек функция. Веднъж когато колбек функцията се изпълни, браузъра прерисува на ново DOM-а и остава в състояние на изчакване за още събития.

Когато браузъра извика Javascript код извън AngularJS изпълнимият контекст, тогава AngularJS не е известен за промяната в модела. За правилна обработка на моделите и промените по тях, всички изчисления трябва да бъдат извършвани във Scope обекта. При такива извън него, AngularJS предоставя \$apply метода, с който направените промени от външен контекст, могат да се отразят правилно върху scope модела и съответстващия му изглед. Ако изчисленията са вътре във scope-а, тогава AngularJS сам си използва \$apply метода и прилага промените.

След изчислението на даден израз, \$apply метода предизвиква \$digest(смилане). Във фазата на смилане, scope обекта преглежда всички изрази които му принадлежат, и които на блюдава, и ги сравнява с предишната им стойност. Тази техника се нарича техника на мръсната проверка (dirty checking). Тя може да бъде доста бавен процес, и за да се избегне забавяне на потребителя – се прави асинхронно.

Интеграция със събитийния цикъл на браузъра



Фиг. 2.2.5.1

Фигура 2.2.5.1 описват начина по който Angular взаимодейства със browser-а.

1. Събитийния цикъл на browser-а изчака за получаване на събитие. За събитие се смята взаимодействието с потребителя, времево събитие(от таймер) или мрежово събитие(отговор от сървъра).
2. Веднъж когато колбак функцията се изпълни, браузъра влиза в Javascript контекста и прерисува наново страницата.

AngularJS предефинира нормалният JavaScript Поток на изпълнения, като предоставя свой собствен цикъл от събития. Това разделя JavaScript-а на класически, и Angular изпълним контекст. Само операции които са приложени във Angular изпълнимия контекст получават предимството на данното свързване. За данново свързване извън Angular контекста – се използва \$apply функцията.

Модули

В AngularJS всяка дефинирана от програмиста функционалност, може да бъде отделена в отделен модул. Към модулите могат да бъдат добавяни филтри, директиви, контролери, услуги(services), и други. Идеята на модулите е да предоставят начин на програмистите да си пренасят лесно кода от едно на друго място. Модулите могат да зависят едни на други, и да използват кода си, като се обяви че даден модул, зависи от други. Въвеждането на даден елемент от модул(директива/филтър/контролер/услуга и т.н.) в друг модул за използване, става чрез Dependency Injection техника.

Инжектиране на зависимости (Dependency Injection)

Инжектирането на зависимости е софтуерен шаблон, при който една, или повече зависимости(или услуги) биват инжектирани, или предадени по референция към зависимият обект (или клиент), и се правят част от клиентското състояние. Шаблонът позволява програмният дизайн да бъде слабо свързан, лесен за промяна и да спазва принципите за единична отговорност. Инжектирането на зависимости включва четири елемента : имплементацията на сървиз обект, клиентски обект зависещ на услугата, интерфейс който клиента използва за да комуникира със услугата, и инжектора, който е отговорен за подаването на услугата на клиента. Инжектор обекта също така може да бъде наричан и асемблер, провайдер, контейнер, фактори(фабрика).

AngularJS имплементира инжектирането на зависимости с помощта на така наречените 'factory functions'. Всеки елемент който бива добавен към модула дефинира в себе си factory function, като в параметрите на тази функция обявява кои са другите елементи, от които той зависи. След като Angular прочете factory function-а, той добавя в списъчна структура срещу името на елемента – списъка със елементите от които той зависи, и при изпълнение на даден код от този елемент, Angular автоматично подава на елемента зависимостите му. По този начин клиентският елемент не е нужен сам да ги инстанцира, и да се грижи за освобождаването им по-късно, тази отговорност се прехвърля на AngularJS.

Пътища

Пътищата в AngularJS са специален вид услуга, която ни дава възможност да променяме пътя в browser-а, да слушаме за дадени събития по пътя, да извличаме параметри от него и т.н. Също както всяка друга услуга в Angular, и пътищата също работят с dependency injection, и за да бъдат включени в клиентския елемент, те трябва да бъдат обявени като част от неговия списък със зависимости, добавяйки \$routeProvider услугата.

Контролери

В AngularJS контролерите са мястото където се съдържа специфичната за приложението логика. Контролерите се използват за запълване на обхват-а(Scope) със всичките необходими данни за изгледа. Използвайки правилно разделение на отговорностите, в контролерите никога не трябва да се съдържа логика която да променя по някакъв начин DOM елементите директно. Тази логика трябва да бъде в директивите.

Контролерите комуникират с изгледа използвайки специална услуга, наречена \$scope. Тази услуга позволява на контролера да даде на обекти и функции на изгледите, които могат по-късно да бъдат манипулирани с изрази или директиви. С цел да бъдат лесно тествани, контролерите са дефинирани чрез dependency injection.

Услуги

Докато във контролерите се съдържа поведението на приложението, което трябва да бъде на разположение на съответния изглед, по някога има нужда от код който да искаме да преизползваме, или който да искаме да абстрахираме от контролерите които говорят директно с изгледите. За тази цел Angular не позволява да дефинираме услуги, които могат да бъдат инжектирани директно в контролерите, или в други услуги.

Ако искаме на пример да комуникираме със сървъра, можем да инжектираме \$http услугата във нашия контролер, или услуга. Друг пример за използването на услуги – е създаването на клас функции за клиентските модели.

2.2.6 Bootstrap

Bootstrap е безплатна колекция от инструменти за създаване на уеб сайтове, и уеб приложения. Тя съдържа HTML и CSS базирани темплейти за дизайн, за типография, форми, бутони, навигация и други интерфейс компоненти, също както и допълнителни JavaScript разширения. През юни 2014 тя е N1 проектът по тегления във GitHub със над 69, 000+ звезди, и 25,000+ разклонения.

Bootstrap е съвместим със последните версии на всички модерни browser-и. Има обратна съвместимост със старите browser-и като IE8. От версия 2.0 поддържа и responsive уеб дизайн. Това означава че изгледът на страницата се наглася динамично, спрямо характеристиките на използваното устройство (настолен компютър, таблет или мобилен телефон).

Bootstrap е с отворен код и е наличен в хранилището GitHub. Разработчиците се окуражават да участват в проекта и да правят техни собствени приноси към платформата.

Структура

Bootstrap е модулярен и се състои от различни Less файлове, които имплементират различните компоненти на инструмента. Стиловата страница наречена bootstrap.less съдържа стиловете страници на отделните инструменти. Разработчиците могат да адаптират Bootstrap файла сам по себе си, като избират кои компоненти биха желали да използват в собствения си проект.

Настройките са налични до определено ниво през централният конфигурационен файл. За по-сложни конфигурации и промени трябва да се променят less декларациите. Използването на езикът за стилизиране .less позволява използването на функции, променливи, вложени селектори, и други. Тези неща ги няма в стандартния CSS, и това прави less-а доста по-мощен от него. Реално Less е спецификация, която бива компилирана до чист .css файл.

От версия 2.0 конфигурацията на Bootstrap също има специална „Customize” опция в документацията. Освен това, разработчика избира през web формата желаните компоненти които иска, и настройва(ако е нужно) стойностите на различните настройки спрямо неговите нужди. Последователно генерираният пакет вече включва предварително направените css стилови страници.

Мрежовата система и responsive дизайнът идват със стандартен 1170 пиксела широт мрежов изглед. Като алтернатива, разработчика може да използва изглед със свои собствени дефиниции за дължините на елементите. И за двата случая

инструментите имат четири различни варианта на дължините, за да могат да се възползват от различните резолюции и типове устройства : мобилни телефони – портретно и пейзажно ориентирани, таблети и компютри с ниска и висока резолюция. Всяка вариация настройва ширините на колоните.

CSS стиловата страница

Bootstrap предоставя набор от стилови страници които предоставят основни стилови дефиниции за всички ключови HTML компоненти. Тези предоставят единен, модерен изглед за форматирането на текст, таблици и формови елементи на цялото web приложение.

Преизползваеми компоненти

В допълнение към нормалните HTML елементи, Bootstrap съдържа други често използвани интерфейс елементи. Тези включват бутони с допълнителни свойства (например групиране на бутони или бутони с падащи опции, навигационни листи, хоризонтални и вертикални табове, странициращи елементи и други), надписи, допълнителни типографски възможности, тъмбнейли, предупредителни съобщения и прогрес бар.

JavaScript компоненти

Bootstrap идва и със няколко JavaScript компонента във формата на jQuery плъгини. Те предоставят на потребителя допълнителни интерфейсни елементи като например диалогови кутии, туултипи и въртящи се галерии. Те също разширяват и функционалността на някои от съществуващите интерфейсни елементи, включващи например само-завършващи (auto-complete) функционалности на входните полета. Във версия 2.0 се поддържат следните JavaScript плъгини : Modal, Dropdown, Scrollspy, Tab, Tooltip, Popover, Alert, Button, Collaps, Carousel и Typehead.

Решетъчна система

В графичния дизайн, решетката е структура (обикновено двуизмерна) съставена от серия от пресичащи се (вертикални и хоризонтални) линии използвана за структуриране на съдържанието. Широко се използва за дизайн на изгледа и съдържанието във принт дизайна. В web дизайна, е много ефективен метод за създаване на консистентен изглед бързо и ефективно, използвайки HTML и CSS. Казано с прости думи решетката в web дизайна организира и структурира съдържанието, правейки уебсайтовете лесни за четене и намаля когнитивните усилия за възприемане положени от потребителите.

В bootstrap е включена решетъчна система състояща се от 12 колони. В нея са включени предефинирани .css класове за лесен избор на изгледа, също както и мощни

добавки за създаване на по семантични изгледи. Решетъчната система се използва за създаване на страници чрез серия от редове и колони които съдържат нашето съдържание. Ето как работи решетъчната система на bootstrap.

- Редовете трябва да бъдат сложени във `.container` клас със подходящо подравняване и отместване.
- Използване на редове за създаване на хоризонтални групи от колони.
- Съдържанието трябва да бъде поставено в колони, и само колони трябва да бъдат преките наследници на редовете.
- Предефинираните класове като `.row .col-xs-4` са налични за бързо съставяне на решетъчни изгледи. Less добавките също могат да бъдат използвани за по-семантични изгледи.
- Колоните създават дупки между съдържанието чрез отместване. Отместването е разстоянието между редовете за първата и последната колона, постигнато чрез отрицателен `margin` на `.rows`.
- Решетъчните колони се създават чрез специфициране на номера на дванайсетте налични ширини които искаме да използваме. Примерно за да използваме три равни колони със ширина 4, трябва да използваме `.col-xs-4`.

Глава 3

Описание на програмното решение

3.1 Описание на клиентската част

Обща информация

Клиентската част на приложението е имплементирана с използването на двата фреймуърка посочени в Глава 2 – AngularJS и Bootstrap. AngularJS е базиран на MVC модела, и чрез него са имплементирани всички интерактивни функционалности на приложението. Bootstrap служи за стилизиране.

Във файловата структура на проекта, клиентските скриптове са разположени във директорията Scripts, като входната точка на AngularJS е app.js файлът.

Рутване, шаблони и изгледи

Тъй като приложението е реализирано като “Single page application” главната заявка към сървъра е само една при началното зареждане на страницата, след това при смяна се прави обръщение към сървъра със Ajax заявка за извличане на нужното съдържание. За реализирането на различните изгледи, и рутването между тях, е използван “ui.router” модулът, който позволява организацията на изгледите на приложението под формата на машина на състоянията. Декларирането на различните състояния на приложението е разположено във app.js. Кодът долу е примерна извадка от настройката на рутването, за да видите цялата конфигурация – погледете сорс файла.

```
RDE.config(function ($stateProvider, $urlRouterProvider, $httpProvider) {  
  $stateProvider  
    .state('Login', {  
      url: '/Login',  
      templateUrl: 'Template/Get/Login'  
    })  
    .state('Project', {  
      url: "/Project/:projectId",  
      templateUrl : 'Template/Get/Editor'  
    })  
    .state('Project.Document', {  
      url: "/Document/:ID",  
      //templateUrl : 'Template/Get/Tinymce'  
    })  
    .state('Project.Diagram', {  
      url: "/Diagram/:ID",  
      //templateUrl: 'Template/Get/DiagramEditor'  
    });  
});
```

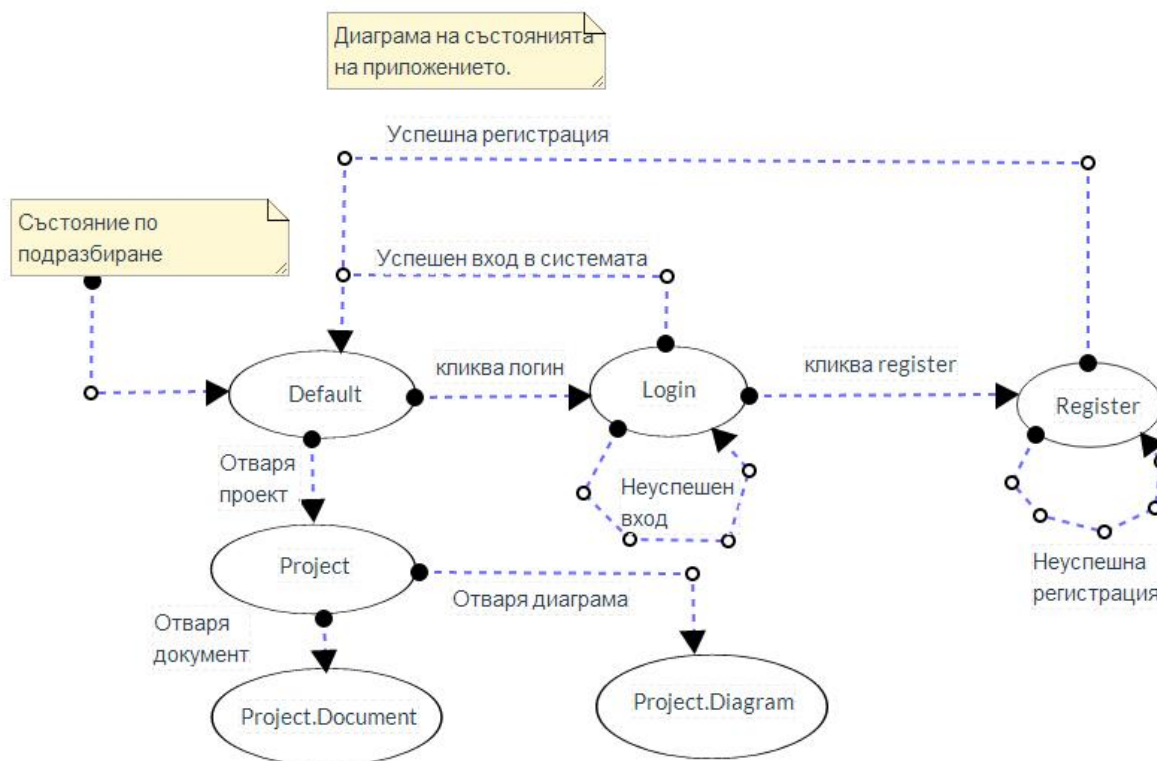
Дефинирането на състояние става чрез подаване на 3 различни параметъра. Първият е име на състоянието, примерно "Project", вторият параметър Url е url-ът на който отговаря това състояние. Третият параметър е templateUrl, при който щом се зареди състоянието – автоматично се зарежда и шаблонът(изгледът) от посоченият адрес. Състоянията могат да бъдат вложени едно в друго, като вложеното състояние следва след името на бащиното състояние – или както е примера "Project.Document" е подсъстояние на Project. Url-ът на вложеното състояние също разширява базовият URL, и по този начин url-ът на състояние "Project.Document" става базовия + наследника, тоест "/Project/:projectId/Document/:ID".

Шаблоните от състоянията се разполагат във div елемента на базовото състояние, и съответно наследяват от обхвата на базовото състояние всички променливи от \$scope обекта. Базовото състояние на всички състояния се дефинира във body tag-a. Там е дефиниран и базовият контролер "BaseController" в който стоят някои основни функции.

```
<body ng-app="RDE">
  <div ui-view class="height-fill mainView" ng-controller="BaseController">
  </div>
</body>
```

Отбелязаните със :ID и :projectId стойности във url-a се явяват параметри, които се променят, в зависимост от желание документ и проект. Те могат да бъдат извлечени по-късно чрез AngularJS функции, за да бъдат използвани за вземане на решения при бизнес логиката.

Диаграма на състоянията на приложението.



Дефинираните шаблони се вземат от посоченият линк на сървъра. Шаблоните представляват чист html код, който бива вложен в страницата от AngularJS. Отговорността за това кой шаблон да бъде върнат на повикването на клиента е възложена на Mvc Controller-ът "TemplateController.cs". В себе си той има дефиниран метод за връщане на шаблона

```
[HttpGet]
public PartialViewResult Get(string name)
{
    return PartialView(String.Format("~/Views/Templates/{0}.cshtml", name));
}
```

Който отговаря на Get заявки, и приема като параметър името на шаблона. След това търси шаблонния файл във /Views/Templates във структурата на проекта, и връща резултата на клиента.

Рутирането от сървърната страна към този контролер е дефинирано във AppStart/RouteConfig.cs

```
routes.MapRoute(
    name: "Template",
    url: "Template/Get/{name}",
    defaults: new { action = "Get", controller = "Template",
name = UrlParameter.Optional }
);
```

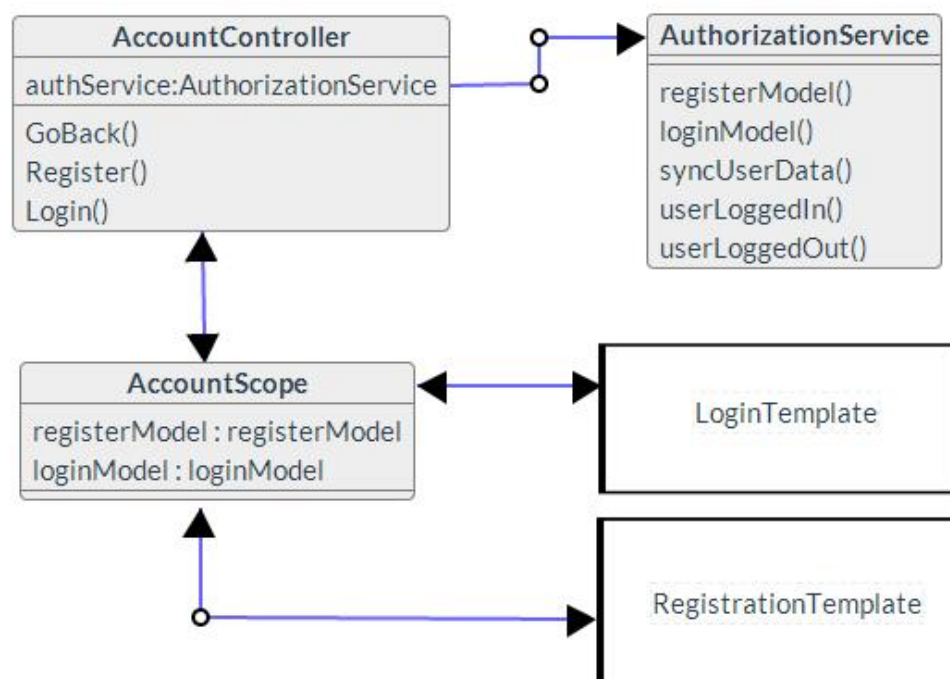
Като тук е описано че всеки път когато към сървъра постъпи обращение "Template/Get/{name}" той ще вземе стойността след Get/ за параметър, и ще я предаде като параметър name на методът Get на template контролера.

Автентикация

Входът в клиентската част съответства на състояние „Login“. В състояние Login се зареждат Login шаблонът, който използва Account контролерът за интеракция с потребителя. Account контролерът от своя страна използва услугите на AuthorizationServices услугата.

За регистрация в системата се използва състоянието "Register". В това състояние се зарежда Register шаблонът. Той също както Login състоянието използва Account контролерът. Обхватът и за двете състояния е един и същи – Account Scope. В него имаме налични два модела – модел за логин, и модел за регистрация. Функциите за логин и регистрация са описани във AccountController-ът.

Следната диаграма показва начинът, по който са обвързани петте обекта.



AuthorizationService обектът е нужен на контролера, защото в него е описана логиката за вход, изход и синхронизиране на данните във системата. Също така – той служи и като фабрика за създаване на login и register обекти при всяко отваряне на Register и Login състоянията.

Пример за използване на AuthorizationService-а в следните извадки от код :

```

$scope.Register = function () {
    $http.post(
        '/api/Account/Register',
        $scope.registerModel
    ).success(function () {
        AuthorizationService.userLoggedIn();
        $state.go("default");
        $scope.serverError = false;
    }).error(function (data) {
        $scope.serverError = true;
        $scope.serverMessage = data.Message;
    });
};

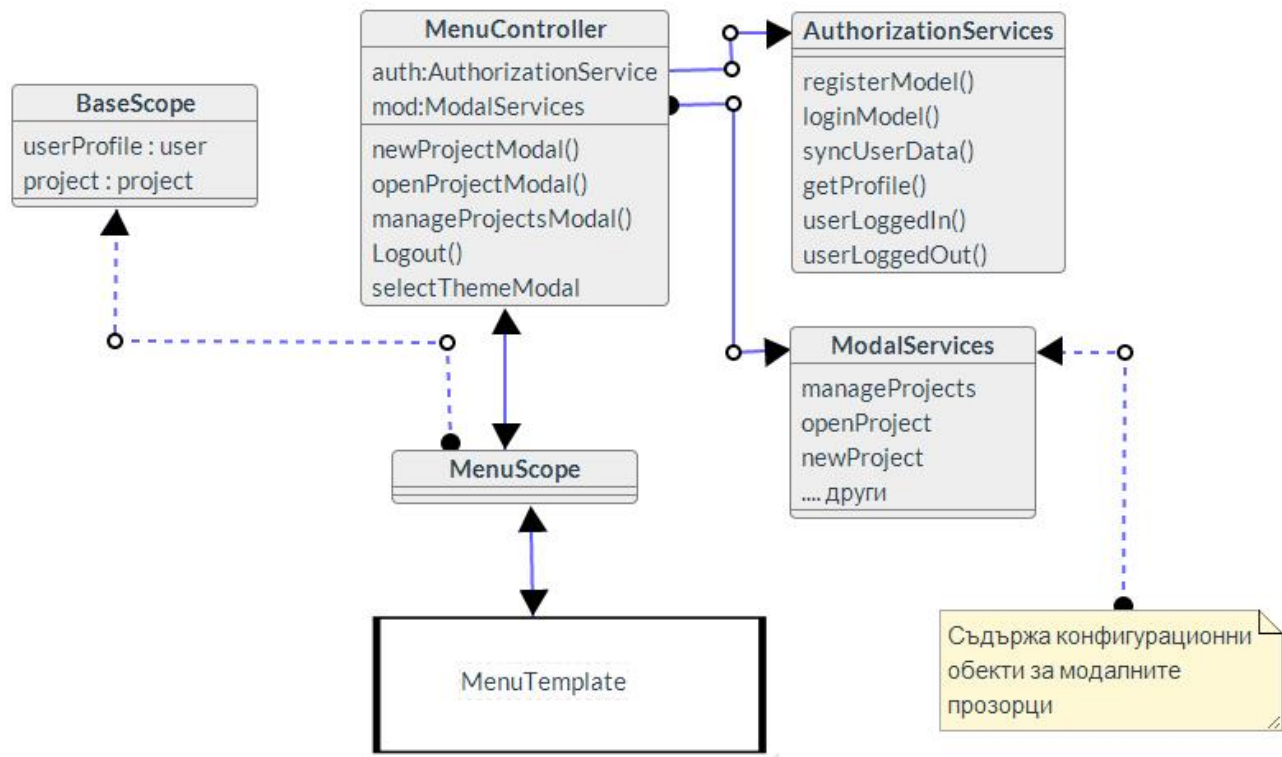
```

Това е регистрационната функция, описана във AccountController-а. Прави се асинхронна заявка към сървъра за вход в системата, като се изпраща попълненият регистрационен модел. При успех се изпълнява методът “userLoggedIn()” на AuthorizationService обектът, който записва че потребителят е влязъл, и прави заявка към сървъра за извличане на неговите данни.

Работа с проекти

Работата с проекти в системата се извършва от падащото меню Project. Налични са 3 опции – Създаване на нов проект, Отваряне на съществуващ и Управление на проектите на текущият потребител.

Организацията на обектите е следната.



В MenuTemplate е дефиниран потребителският интерфейс. От връзката си към MenuScope-а, той прави връзка с MenuController, където са разположени всички интерактивни функции. Примерен код за връзката между MenuTemplate и MenuScope имаме в следният ред код.

```

<li><a ng-class="{ 'disabled-link' : !userProfile.logged}"
      ng-click="newProjectModal()"><span class="glyphicon glyphicon-briefcase"></span> New
project</a></li>
<li><a ng-click="openProjectModal()"><span class="glyphicon glyphicon-folder-open"></span> Open
project</a></li>

```

Тук имаме дефинирани два бутона – съответно за създаване на нов проект, и за отваряне на съществуващ проект. Атрибутът ng-click се явява вградена директива на

AngularJS, която извиква функцията `newProjectModal`. Тази функция се търси във `$scope` обекта, който има връзка към `Menu` контролера. От там функцията се изпълнява.

```
$scope.newProjectModal = function () {  
    if ($scope.userProfile.Logged)  
        $modal.open(modalServices.newProject);  
};  
  
$scope.openProjectModal = function () {  
    $modal.open(modalServices.openProject);  
};
```

Функциите имат обръщение към `$modal` услугата на AngularJS библиотеката, която създава нов модален диалог. За създаване на нов диалог, е нужно да бъде подаден конфигурационен обект на диалога, който `MenuController`-а взема от `modalServices` обекта. В `ModalServices` обекта предоставените обекти съдържат само настройки на диалозите.

```
var newProject = {  
    templateUrl: '/template/modal/newProject',  
    controller: 'NewProjectController',  
    backdrop: 'static'  
};  
  
var openProject = {  
    templateUrl: '/template/modal/openProject',  
    controller: 'OpenProjectController',  
    backdrop: 'static'  
};
```

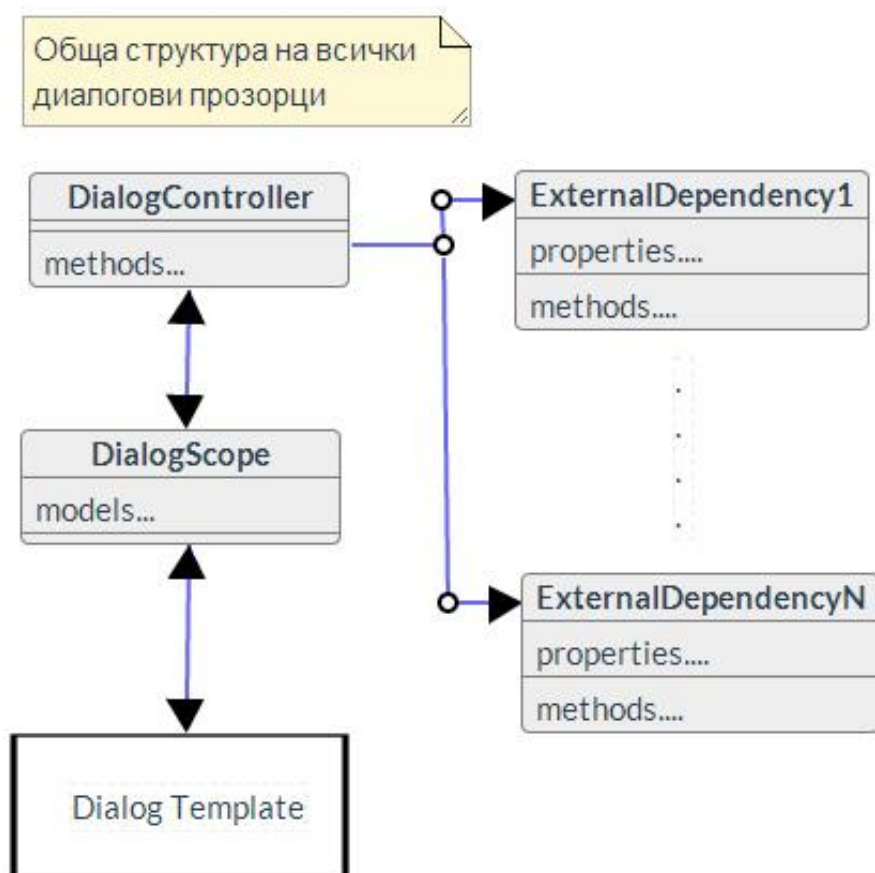
Настройките включват :

- `templateUrl` – Връзка към шаблона на диалога. Прави се обръщение към сървъра.
- `Controller` – Контролер на диалога, който е отговорен за взаимодействието с него.
- `Backdrop` – Указва дали фоновите елементи да бъдат активни, когато е активен даденият контролер.

Диалогови прозорци

В приложението са налични 6 диалогови прозореца, всеки с различна задача, дизайнът на всички диалогови прозорци в приложението е един и същ. Състои се от шаблон на диалоговият прозорец, обхват на диалоговият прозорец, контролер който отговаря за интеракцията с потребителя, и външния зависимости (услуги на приложението) на диалоговият прозорец, които се инжектират във конструктор функцията на контролера.

Примерна схема на връзките между елементите на диалоговите прозорци.



Dialog Template елементът представлява HTML изгледът, който служи за визуализиране на съдържанието на диалоговия прозорец. DialogScope е обхватът, в който са разположени моделите, които ще се използват от диалога, dialogScope-а служи за свързването на данните между шаблона и модела. В DialogController са описани методите на диалоговите прозорци. Всички външни зависимости (ExternalDependency1 – ExternalDependencyN) се реферират от диалоговия контролер, и се използват техни методи, или свойства в дефинираните в контролера методи.

Клиентски услуги

До тук вече няколко пъти бяха споменавани различните услуги, които се използват на клиентската част. Под услуги в контекстът на AngularJS се разбира обект с глобална видимост, който може да бъде инжектиран във всеки друг самостоятелно дефиниран елемент на AngularJS(друга услуга, контролер, директива, или филтър).

Списък с услугите на клиентската част:

- AuthorizationService – съдържа методи отговарящи за комуникацията със сървъра свързани с автентикацията на потребителя.
 - syncData() – синхронизиране на данните на потребителския профил.
 - userLoggedIn() – отбелязва потребителя като влязъл, и синхронизира данните.
 - userLoggedOut() – отбелязва потребителя като излязъл, и трие потребителската информация.
 - getProfile() – връща потребителския профил
 -
- ModalService – съдържа конфигурационни обекти на диалоговите прозорци.
- ErrorHandlingService – Съдържа методи за отработване на грешките в резултат на комуникацията със сървъра. Грешки е възможно да възникнат при грешно зададени параметри на изпращаните данни към сървъра, примерно когато потребителя се опита да модифицира някой файл от проекта, но не е част от този прокт, или се опита да запази съдържанието на файл, но му е изтекла потребителската сесия. ErrorHandlingService преглежда кодът за грешка от сървъра, и зависимост от кода – изкарва различно съобщение в отделен модален прозорец.
- ProjectService – Съдържа логика за отваряне/затваряне и модела на проекта.
 - loadProject(id) – Прави обръщение към сървъра за извличане на модела на проект по зададено ID на проекта.
 - unloadProject – Маркира проекта като не зареден.
 - Project – моделът на проекта.

Интересно нещо за отбелязване при ProjectService-а е начинът, по който той превръща линейната структура на проекта в дървовидна. След заявката към сървъра за клиентският модел на проекта, сървърът връща линейна структура със артикулите в проекта. Тъй като проекта съдържа в себе си папки, то те могат да се влагат едни в други, но трябва да се извърши преобразуване на данните, за да се създаде дървовидната структура.

Алгоритъмът работи чрез 2 цикъла. След края на първия цикъл се създава речник, който съдържа ID-тата на всички папки като ключ, и празен списък като стойност. След края на втори цикъл се проверява дали артикулът има ParentID, ако има, то от речникът се взема списъкът в който трябва да се сложи, и се премахва от първичната линейна структура.

```
var createTreeStructure = function () {  
    var folders = {};  
    for (var i = 0 ; i < project.Items.length ; i++)  
    {  
        if (project.Items[i].Type == ItemType["Folder"])  
        {
```



```
        var items = [];  
        project.Items[i].Items = items;  
        project.Items[i].opened = true;  
        folders[project.Items[i].ID] = items;  
    }  
}  
  
for(var j = 0 ; j < project.Items.length ; j++)  
{  
    var item = project.Items[j];  
    if (item.ParentID != -1)  
    {  
        folders[item.ParentID].push(item);  
        project.Items.splice(j, 1);  
        j--;  
    }  
}  
};
```

- ToolsService – Съдържа логика за работа с инструментите на едитора на диаграми. Ще бъде разгледан по-обстойно по-късно.

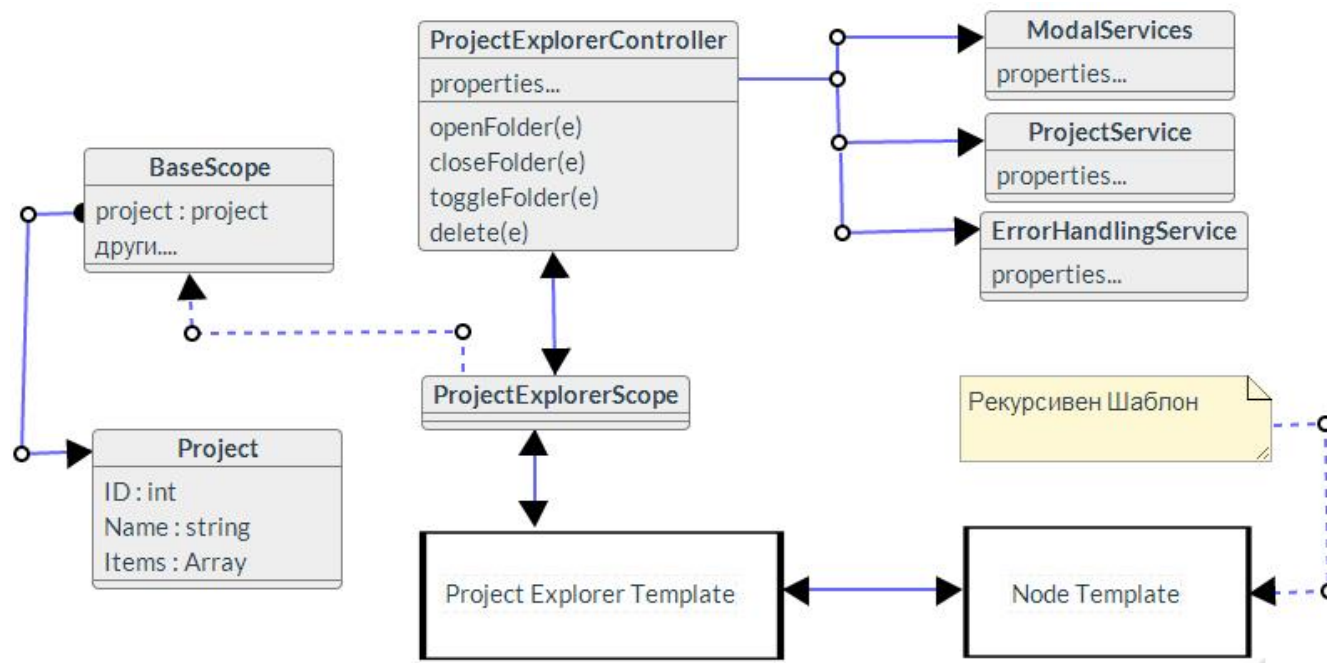
Стилизиране на приложението

За стилизирането на приложението е използван Bootstrap framework. Bootstrap предоставя на програмиста предварително създадени .css файлове, от които той избира нужните му класове. Bootstrap предоставя 16 безплатни предефинирани теми. Всяка тема представлява отделен .css файл. Класовете във всичките 16 файла са еднакви, и единственото нещо което се сменя при смяната на файловете – са цветовете и шрифтовете.

Смяната на темите на приложението е реализирана чрез използване на бисквитки(cookies). В бисквитка се записва коя е желаната тема, която потребителят иска да използва, след това при обръщението си към сървъра, той отговаря със съответният .css файл на темата. Логиката за избиране на тема и записване в cookie е описана във SelectThemeController.js файлът.

Project explorer

Project Explorer-а на проекта има за цел да предостави на потребителя възможност да прегледа цялостната структура на проекта, да добавя/изтрива/отваря артикули от проекта. Структурата на обектите от които е съставен Project Explorer-а можете да видите на следващата диаграма.



В ProjectExplorerController-a е разположена главната логика на Project Explorer-a. Той от своя страна реферира ModalServices ProjectService и ErrorHandlingService за да използва техните методи и свойства за собствената си бизнес логика. ProjectExplorerController-a реферира ProjectExplorerScope-a, който наследява базовият BaseScope. В BaseScope-a имаме модела на проекта, който съдържа в себе си ID на проекта, име и списък с артикули в себе си.

От ProjectExplorerScope-a модела се предава на ProjectExplorerTemplate-a където информацията се визуализира. Важно значение тук има рекурсивният шаблон NodeTemplate.

```

<ul ng-show="node.opened">
  <li
    ng-repeat="node in node.Items"
    ng-include="'/Template/Get/Node'"></li>
</ul>

```

Този код е извадка от Node шаблонът. В себе си той повтаря със ng-repeat директивата на AngularJS всички артикули които се намират в списъка на подадения възел (това е само в случай че възелът е папка) и подава артикулите на артикула на същият този шаблон.

```

<span
  class="pointer"
  ng-dblclick="(node.Type==1)?toggleFolder(node):openItem(node)"
  ng-context-
  menu="(node.Type==1)?folderOptions:(node.Type==2)?documentOptions:diagramOptions"

```

```
ng-context-elem="node">
<span
  class="glyphicon"
  ng-class="
    {'glyphicon-folder-open': node.Type==1 && node.opened,
    'glyphicon-folder-close': node.Type==1 && !node.opened,
    'glyphicon-file': node.Type==2,
    'glyphicon-picture': node.Type==3}"></span> {{node.Name}}
```

В тази извадка от код се дефинира логиката на всеки един възел в ProjectExplorer-a. Със ng-dblclick директивата се описва поведението на възела при двойно кликане. Ако възелът е от тип папка – то тя се отваря/затваря, ако е от тип диаграма/документ – то тя се отваря. Всеки възел също така има налично контекстно меню чрез ng-contextmenu директивата. Опциите на контекстното меню са описани във ProjectExplorerController-a.

Текстов редактор

При работа с документи потребителят има възможност за използване на текстов редактор, за обработката на текста. Текстовият редактор използван в текущият проект е TinyMCE. TinyMCE е платформи независим web-базиран Javascript/HTML WYSIWYG(what you see is what you get) едитор, с отворен код под LGPL лиценза. Той има свойството да конвертира HTML текстови полета или други HTML елементи към инстанции на едитора. TinyMCE е направен с цел лесно интегриране със среди за редактиране на съдържанието, като Django, Drupal, Joomla, Wordpress и други.

За да се интегрира подходящо с текущият проект, е използван модул, на който текущият модул на приложението зависи. Името на модула е “ui.tinymce”. В него се декларира директива за създаване на нова инстанция на TinyMCE редакторът.

В tinyMCE директивата се подава конфигурационен обект options, който се прилага върху зададения от директивата елемент, и така се създава инстанция на едитора. Важно да се отбележи е че при създаването на директивата се използва ng-model контролерът на AngularJS, с който текстът във TinyMCE се обвързва със модела, подаден на ng-model, и по този начин – щом имаме промяна в текстовият редактор, то тя се появява директно и във модела, и обратното. По този начин се осъществява двустранното свързване (data-binding) между модела на tinyMce директивата, и инстанцията на едитора.

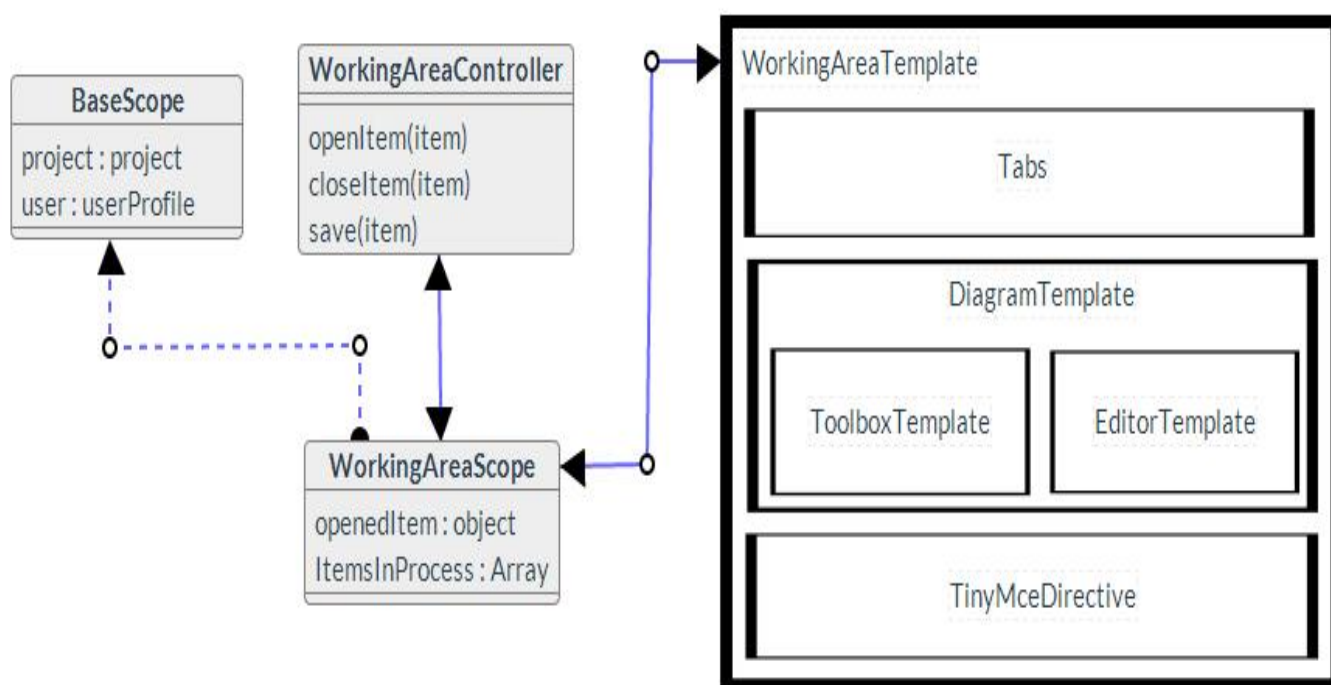
Примерен код за използването на декларираната директива можете да видите на следващите редове.

```
<div ng-show="openedItem.ref.Type==2">
  <textarea id="tinymce" ui-tinymce ng-model="openedDocument.dirtyContent"></textarea>
</div>
```

Тук текстовата area се декорира със ui-tinymce директивата, бива и подаден модел – моделът на отвореният документ.

Работна зона

Обектите участващи в реализацията на работната зона са WorkingAreaController, WorkingAreaScope BaseScope и WorkingAreaTemplate. Връзката емжду компонентите може да видите на следващата диаграма.



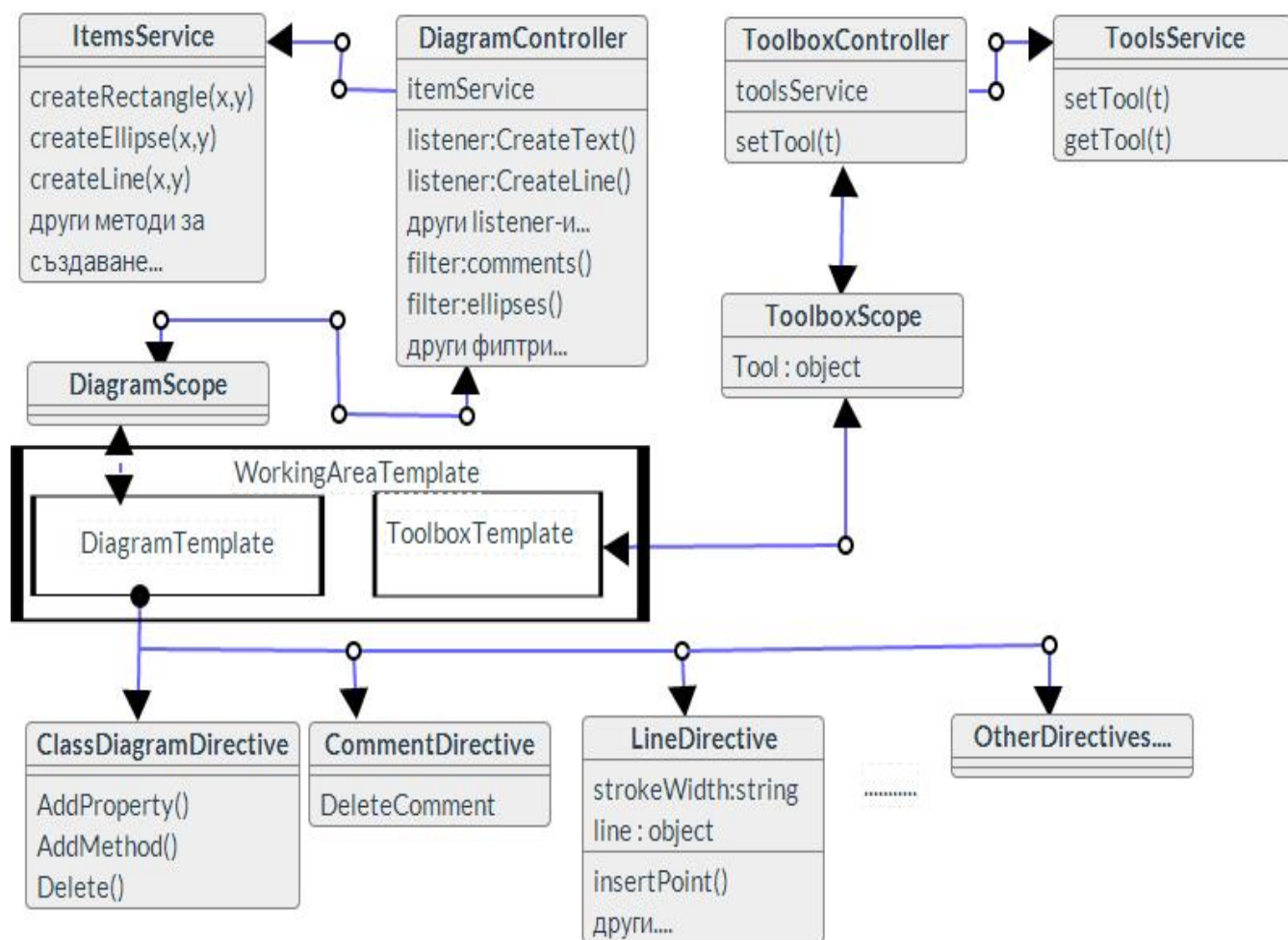
WorkingAreaTemplate-а отговаря за визуализирането на информацията. В себе си той съдържа 3 основни елемента.

- Tabs – табове служат за организирането на отворените документи във табуларна структура. Смяната между табове се извършва с кликане. Събитието което отговаря на кликане е смяна на състоянието на приложението.
`ui-sref="Project.Document({ID:item.ref.ID, projectId : project.ID})"`
 Приложението тогава в състояние в зависимост от типа на таба, като се подават ID-то на артикула, и ID-то на проекта.
 Логиката която отработва смяната на състоянието стои във WorkingAreaController-а.
`$scope.$on('$stateChangeSuccess', function() { ... }).` Там на базата на

подадените параметри се взема типът на артикула, и в зависимост от него се избира кой от двата елемента “DiagramTemplate” или “TinyMceDirective” да бъде показан на потребителя.

- DiagramTemplate – Това е шаблонът отговорен за показване на диаграмният редактор на потребителя. В себе си той съдържа Toolbox – за инструменти, и Editor – за реалната работна област на редактора.
- TinyMceDirective – Съдържа в себе си инстанция на TinyMCE редакторът за редакция на документи, за повече информация за директивата – погледнете по-горе нейното описание.

Едитор на диаграмите



Едиторът на диаграмите има за задача да даде възможност на потребителя да създава, редактира и изтрива диаграми. Той се състои от 2 основни изгледа – DiagramTemplate и ToolboxTemplate.

ToolboxTemplate Има за цел да покаже на потребителя наличните инструментни. Обработка се от ToolboxController-ът, който има една единствена възможна функция – setTool(t), с която се обръща към ToolboxService-а, и задава стойност на избрания Tool. Спрямо избрания Tool във ToolboxService-а се поставя различна стойност на click функцията на DiagramTemplate-а. Реализирано е чрез един голям Switch, с който се проверява подадената стойност от контролера.

```
switch (t) {  
    ...  
    case "ZoomIn": {  
        tool.click = zoomInClick;  
        break;  
    }  
    case "ZoomOut": {  
        tool.click = zoomOutClick;  
        break;  
    }  
    case "ClassDiagram": {  
        tool.click = createClassDiagram;  
        break;  
    }  
    ...  
}
```

Във ToolsService-а има дефинирани функции за всички tool-ове като частни променливи, недостъпни от външният свят. Начинът по който работят Tool функциите е чрез broadcast-ване. При клик се broadcast-ва към всички scope-ове на приложението че е кликнато върху едитора, и подходящия контролер обработва действието. Архитектурата е избрана по този начин с цел – по лесно разширение по-нататък.

Listener-ите за създаване на обекти са дефинирани във DiagramController-а. Там при постъпване на събитие, контролера създава обекта съответстващ на събитието, и го добавя към widget-ите на отворената диаграма. Логиката за създаването на отделните модели е изкарана във ItemsService услугата.

След като са добавени към списъкът със widget-и на диаграмата, моделите биват изрисувани на екрана във DiagramTemplate изгледа.

```
<class-diagram ng-repeat="diagram in openedDiagram.dirtyContent.widgets | filter:classDiagrams"  
diagram="diagram"></class-diagram>  
<comment ng-repeat="comment in openedDiagram.dirtyContent.widgets | filter:comments"  
comment="comment"></comment>  
<line points="line.points" line="line" ng-repeat="line in openedDiagram.dirtyContent.widgets |  
filter:lines"></line>  
<text text="text" ng-repeat="text in openedDiagram.dirtyContent.widgets | filter:texts"></text>
```

```
<rde-ellipse model="ellipse" ng-repeat="ellipse in openedDiagram.dirtyContent.widgets |  
filter:ellipses"></rde-ellipse>  
<rde-rectangle model="rectangle" ng-repeat="rectangle in openedDiagram.dirtyContent.widgets |  
filter:rectangles"></rde-rectangle>
```

Както се забелязва – дефинираните във изгледа HTML тагове не са стандартни тагове. Всички тези тагове представляват предефинирани директиви, затова template-a има връзка към всичките директиви, които са за отделните widget-и на едитора.

Директивите са самостоятелни функционалности, които могат лесно да бъдат прехвърляни от едно приложение на друго. Всяка директива сама по себе си притежава (опционално) контролер, изглед и други атрибути. Принципът на работа с всички е един и същи. За да не разглеждаме всичките 8 директиви поотделно, ще дам за пример само ClassDiagram директивата.

ClassDiagram Директива

Класдиаграм директивата е самостоятелна функционалност, която позволява създаването на отделни class widget-и. Във контролерът на директивата имаме описани отделни функции, които тя може да изпълнява, като например:

```
$scope.deleteMember = function (e) { .... } – изтриване на член  
$scope.addProperty = function () { ... }; - добавяне на пропърти  
$scope.addMethod = function () { ... }; - добавяне на метод  
$scope.deleteDiagram = function () { ... } – изтриване на целия  
class widget
```

На директивата също така и се подава и Шаблон, с който да се изобрази class widget-ът. Angular прави асинхронно обръщение към сървъра за вземане на шаблона.

След поставянето и във DOM-а, AngularJS поставя на мястото на директивата нейният шаблон, и записва методите на контролера, към клик събитията на посочените в шаблона елементи. Останалите 7 директиви работят по подобен начин.

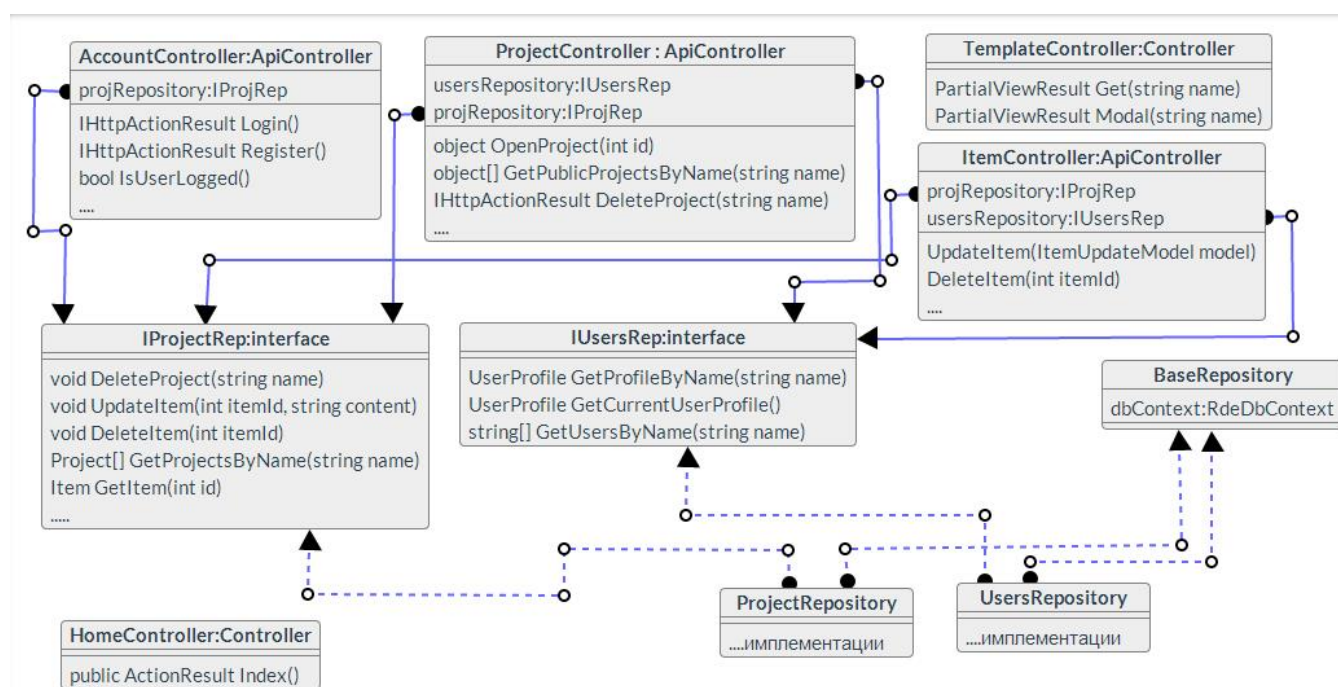
3.2 Описание на сървърната част

Сървърната част на приложението е реализирана с помощта на 3 технологии : Asp.Net MVC, Asp.Net Web Api и EntityFramework.

За комуникация с базата данни, се използват хранилищата(Repositories), които наследяват базовото хранилище, в което има дефиниран dbContext обект за връзка с базата данни. Хранилищата се достъпват от контролерите през интерфейси. Целта на това абстрахиране е с цел – по късно ако се пишат unit тестове, да може лесно да се заменят хранилищата към базата данни, със Mock обекти за симулиране на връзка, и по-бързо изпълнение на unit тестовете.

Налични са 2 вида контролери – ApiController и Controller. ApiController-ите са тези, които са предоставени от WebApi framework-а, тяхната задача е да отговарят на асинхронните поисквания на клиента, и да връщам JSON резултати, докато задачата на Controller-ите е да отговарят на запитванията за клиента, които са свързани със поискване на HTML съдържание.

Структурата на сървърната логика можете да видите на долната каритнка.



Контролери

Входната точка на приложението се пада `index()` методът на `HomeController`-а. От там на клиента се връща главния изглед на приложението, със всички скриптове и стилове в него.

За автентикация и вход в системата, асинхронните заявки от клиента (описани във `AuthorizationService`-а на клиентската част) се обръщат към `AccountController`-а, чиято задача е да осъществява вход, изход, информация за профила на потребителя и други. За регистрация на потребителя, контролера трябва да направи заявка за записване на профила в базата. За тази цел той им референция към `IUsersRepository`-то.

`ProjectController`-а има задача да отговаря на заявки свързани със запитвания за проекта от клиента. При заявка за отваряне на проект, контролера отговаря със следния метод :

```
public object OpenProject(int id)
{
    var proj = projectRepository.GetProject(id);
    proj.Items.Sort();
    return new
    {
        ID = proj.ID,
        Name = proj.Name,
        Items = proj.Items
    };
}
```

Връщаният тип е "object" защото не е дефиниран ясен модел, който да се върне на потребителя. `Return new { ... }` е анонимен обект, за който след компилация се създава клас. Реално този тип е дефиниран с клас, но това се извършва от компилатора за нас автоматично. Целта на този синтаксис и на това записване и връщане на моделите е, по-голяма гъвкавост, и по-лесен за промени код по време на разработка, докато не са на 100% ясни всичките стойности, които модела трябва да притежава. Възможно е в бъдеще да се замени със силно типизирана инстанция на даден клас, с цел подобряване на качеството на кода.

За отговаряне на заявките на клиента за запазване на отделни артикули в проекта, задачата се възлага на `ItemsController`-а. Характерно за този контролер е че той следи нивата на достъп до поискания артикул, като за отделната операция проверява дали потребителя който я извършва има зададените права.

Примерен код е следващият фрагмент :

```
var user = System.Web.HttpContext.Current.User.Identity.Name;
```

```
var proj = projectRepository.GetProject(projectId);

var participation = proj.Participations.Find(x =>
x.UserReference.UserName == user);
if (participation != null
    && participation.AccessLevel >= Accessibility.Read_Edit_Create
    || participation.AccessLevel == Accessibility.Owner)
```

Взема се текущият user, след това се взема проекта, в който е item-а, проверява се дали участието на потребителя във текущия проект е по-голямо или равно на нивото за създаване на артикули, или дали потребителя не е създателят на проекта.

Друг важен контролер за системата е TemplateController-а. Както беше описано в клиентската част на приложението има много асинхронни заявки за частичен HTML резултат. Заявките идват от шаблоните за диалоговите прозорци, от директивите, и от вложените условни изгледи. TemplateController-а дефинира 2 метода за връщане на частични резултати.

```
[HttpGet]
public PartialViewResult Get(string name)
{
    return PartialView(String.Format("~/Views/Templates/{0}.cshtml", name));
}

[HttpGet]
public PartialViewResult Modal(string name)
{
    return PartialView(String.Format("~/Views/Templates/Modals/{0}.cshtml",
name)); }
```

И двата метода отговарят само на Get заявки, единият е отговорен за връщането на модалните шаблони, другият – за връщането на всякакъв друг вид шаблони. Разделението е направено с цел – по-лесна организация, и ориентиране в структурата на проекта.

Хранилища

За връзка с базата данни се използва EntityFramework. Хранилищата там са 2 – UsersRepository и ProjectRepository. И двата класа наследяват Base repository, където като частен член е дефиниран контекстният клас RdeDbContext на приложението, във структурата на контекстният клас са разположени 5 DbSet-а за връзка с различните таблици към базата данни.

```
public partial class RdeDbContext : DbContext
{
    public RdeDbContext()
        : base("name=RdeConnectionString")
    {
    }
}
```

```
public DbSet<UserProfile> UserProfiles { get; set; }  
public DbSet<Project> Projects { get; set; }  
public DbSet<Participation> Participations { get; set; }  
public DbSet<Item> Items { get; set; }  
public DbSet<Template> Templates { get; set; }  
}
```

Контекстът се създава като се вземз connectionString-а от настройките на Web.config-а на приложението. След създаването на контекста, при обръщение към коет о и да е от 5те DbSet property-та, EntityFramework следи за промени, и автоматично превежда промяната към SQL заявка, и я изпраща към базата данни.

EntityFramework се явява доста бързо и лесно решение за извършване на елементарни операции в базата данни, защото кодът се пише само едностранно от програмиста, след това базата автматично си прави съпоставките, промените, и записите в базата данни.

Двете хранилища имплементират интерфейси, съответстващи на операциите които трябва да извършват, през които да бъдат достъпвани. Разделението се прави с цел – ако по-късно се правят unit тестове да може лесно да бъдат заменени с mock обекти, за бързо изпълнение на тестовете.

3.3 Описание на базата от данни

Базата от данни за системата е разработена на MS SQL сървър и следва да отговаря на следните условия :

- Да съхранява всички данни за проекта.
- Да съхранява потребителските данни.
- Да съхранява данни за артикулите в проектите.

Определяне на наборите обекти

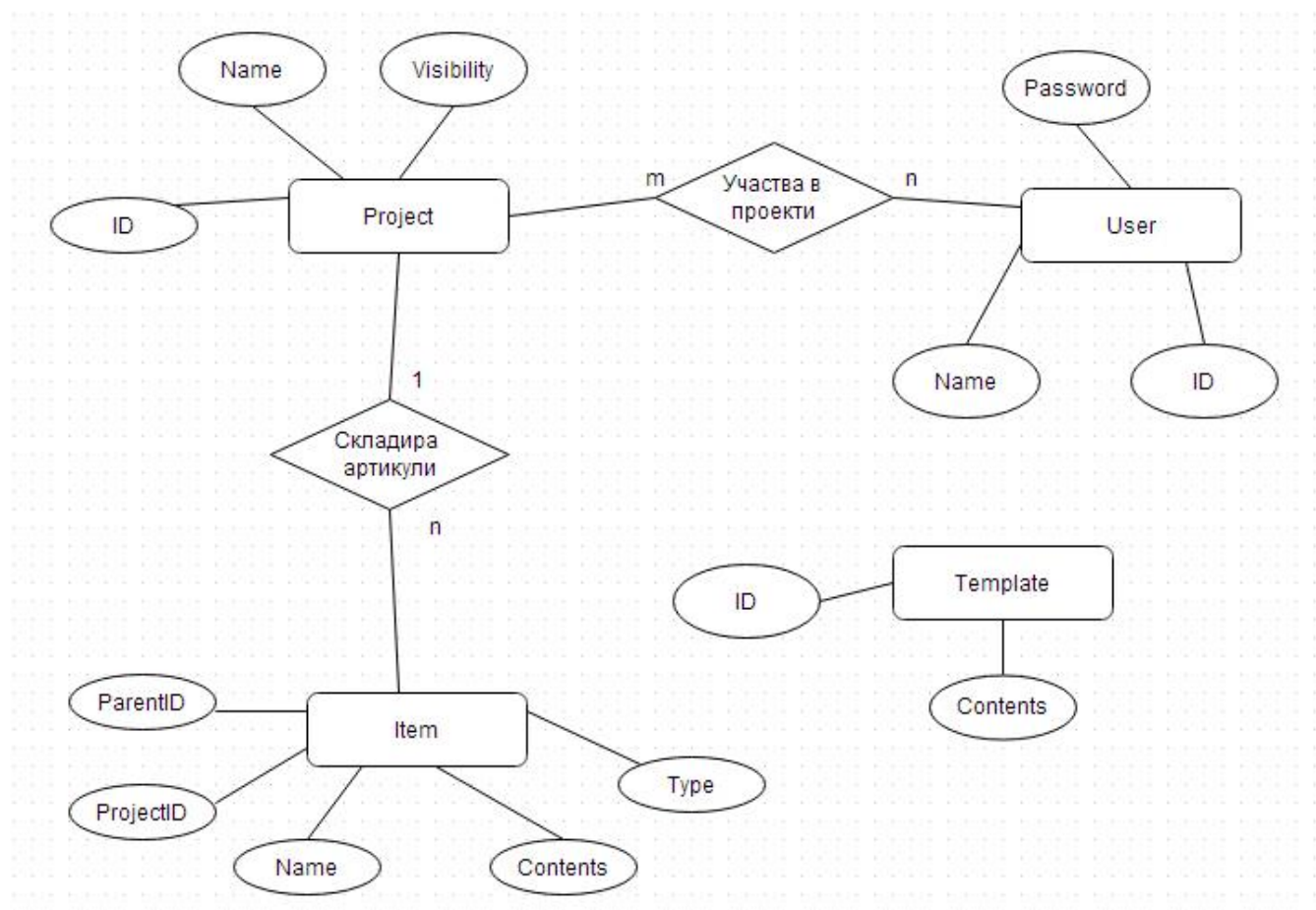
Варианта на базата данни за системата за дизайн и анализ на изискванията се състои от 4 основни таблици, които осигуряват правилната работа на системата.

- Информация за проекти – съдържа всичката нужна информация за даден проект. Включително – ID, Име, Видимост на проекта.
- Информация за Артикулиет в проектите – Съдържа информация за ID на артикула, име на артикула, ID на бащата на артикула, ID на проекта на артикула, тип на артикула, съдържание на артикула.
- Потребителски профил – съдържа информация за името на потребителя, ID на потребителя, парола на потребителя и др.
- Шаблони на артикули – съдържа информация с предварително дефинирани стойности, когато потребителя реши да създаде артикул по шаблон, пример за това е – създаването на документ с изискванията при съставяне на нов документ. Нужна информация – ID на шаблона, съдържание на шаблона.

При проектирането и реализацията на релационната база от данни процесът на нормализация има следните практически цели :

- Да се освободи наборът релации от излишъци и нежелани тойности при корекции и зитриване на информация от БД.
- Да се избегне необходимостта от прегрупиране на набора релации при въвеждането на нови типове данни, което позволява да се удължи живота на приложните програми.
- Да се изтрият повтарящи се атрибути, в съседните таблици, за да не се натоварва базата с данни.

Универсално представяне на структурата на базата данни.



Първа нормална форма

Премахват се повтарящите се атрибути. Една релация се намира в първа нормална форма тогава и само тогава, когато всички влизаци в нея домейни съдържат само атомарни стойности.

Втора нормална форма

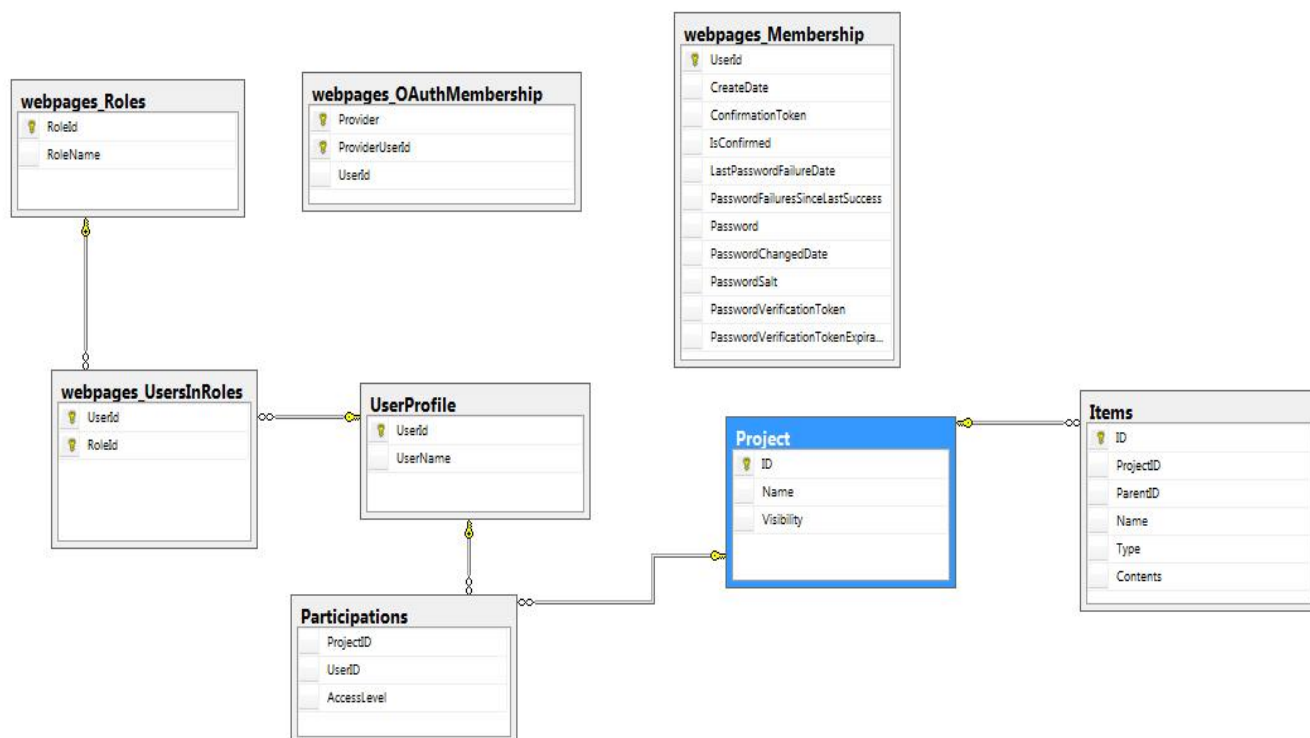
Една схема е във втора нормална форма ако се намира в първа нормална форма и всеки не първичен атрибут изцяло зависи от всеки ключ за схемата на отношението.

Трета нормална форма

Една релация е в трета нормална форма, ако вече е в първа нормална форма и всеки непървичен атрибут, не зависи транзитивно от всеки възможен ключ. За да бъде

изпълнено това условие, са разлизирани отделни отношения, така че в една таблица да съществува единствено функционална зависимост не ключови и ключови атрибути.

Реална структура на базата данни.




Описание на таблиците

- Таблица Project.

	Column Name	Data Type	Allow Nulls
	ID	int	<input type="checkbox"/>
	Name	nvarchar(225)	<input type="checkbox"/>
	Visibility	int	<input type="checkbox"/>

Тази таблица съдържа информация за проектите, които се създават във системата. Всеки проект има ID, име и видимост. Името на проекта е от тип Varchar, видимостта е от тип Byte, видимостта съответства на енумерацията на свързната страна ProjectVisibility. ID – уникален ключ, по подразбиране се инкрементира с 1 от базата данни.


- Таблица Items

	Column Name	Data Type	Allow Nulls
	ID	int	<input type="checkbox"/>
	ProjectID	int	<input type="checkbox"/>
	ParentID	int	<input type="checkbox"/>
	Name	nvarchar(50)	<input type="checkbox"/>
	Type	int	<input type="checkbox"/>
	Contents	nvarchar(MAX)	<input checked="" type="checkbox"/>

Съдържа информация за артикулите които принадлежат към проектите. ProjectID-то на таблицата е foreign key към таблицата Project. Връзка 1 : m. Съдържанието в таблицата се записва под формата на JSON стринг. След като се подаде на клиента там се парсира и се работи с него под формата на javascript обекти, след което се праща отново на сървъра като JSON стинг и се пази в таблицата. ParentID описва родителя на артикула, като по този начин позволява влагане на артикулите един в друг.

- Таблица UserProfile Съдържа информация за потребителя, потребителско име и ID.

- Таблица Participations

	Column Name	Data Type	Allow Nulls
	ProjectID	int	<input type="checkbox"/>
	UserID	int	<input type="checkbox"/>
	AccessLevel	int	<input type="checkbox"/>

В тази таблица се извършва връзката между проекта, и потребителите които участват в него. Допълнителната колона "AccessLevel" дава възможност за задаване на нивото на достъп на потребителя до зададения проект, съответства на сървърната енумерация AccessLevel. Двете колони ProjectID и UserID са foreign key-ове към таблиците Project и User.

- Таблица Templates – в тази таблица се пазят предварително дефинирани шаблони за артикулите създавани по шаблони.

- Служебни таблици – Служебните таблици започващи с префикса "webpages" са таблици на автентикационния провайдер използван в системата (Simple membership

provider), **тези таблици се създават от provider-a, и той сам определя тяхната структура.**

- webpages_UsersInRoles – За връзка между роли и потребители, не се използва в настоящия проект.
- webpages_Roles – За дефиниране на роли, не се използва в настоящия проект.
- webpages_OauthMembership – За използване на автентикация чрез OAuthProvider, не се използва в настоящия проект.
- webpages_Membership – За запазване на информация за потребителя от SimpleMembershipProvider-a. В тази таблица се пазят данни като – хеш на паролата, брой неуспешни влизания (с цел защита), дата на създаване на акаунта и други.

Глава 4

Ръководство на програмиста

4.1 Инсталиране на системата

В случай че екипът от програмисти които желаят да използват системата искат да хостват собствено копие, което да върви на техни машини локално, за да имат контрол над данните в базата, ще имат нужда от инсталиране на следните приложения на сървърната машина :

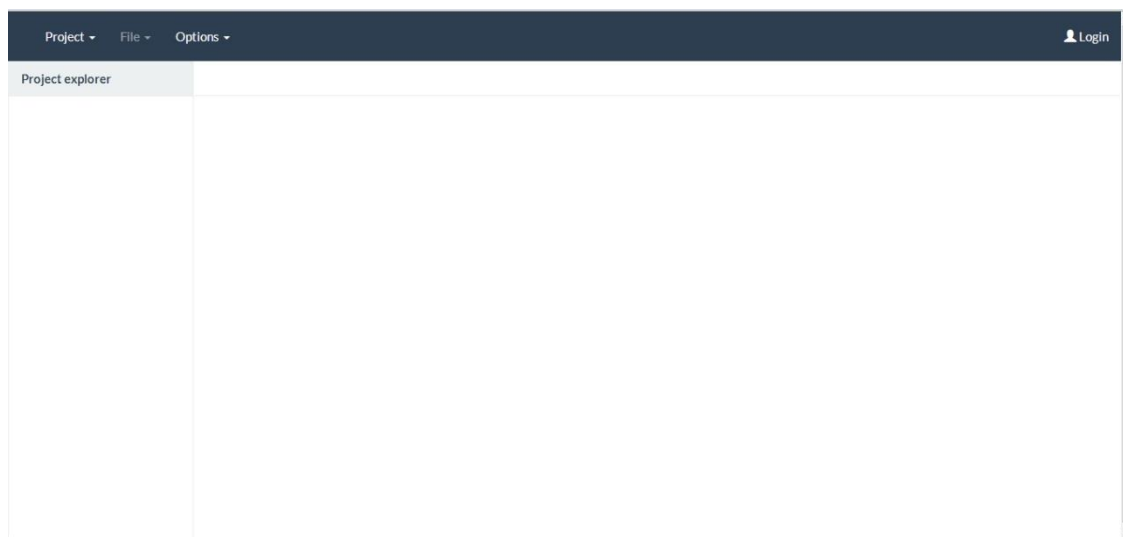
- .Net Framework ver. 4.5 или по-нова.
- Internet Information Services ver. 7 или по-нова.
- Microsoft SQL Server 2008 или по-нова версия.

В случай че горе изброените приложения са налични, моля следвайте следните стъпки за успешно инсталиране на системата.

1. Отворете IIS Manager-а. Кликнете върху старт менюто на windows-а, в полето за търсене напишете IIS Manager.
2. В ляво на панелът с конекциите (connections) кликнете върху Sites с десен бутон и кликнете Add Website.
3. В появилият се прозорец въведете
 - 3.1. Име на сайта в полето Site name
 - 3.2. Физически път в полето physical path, или го изберете от бутона.
 - 3.3. Номер на портът на който сървърът да обработва заявките (препоръчително е да го оставите на 80 порт).
4. След като изпълните тези операции ще видите в ляво името на създаденият от вас сайт. За да гостартирате – кликнете върху него с десен бутон и кликнете върху бутона “Start Site”
5. За да установите връзка със базата данни отворете физическият път на приложението (възможно е чрез дясно кликане върху сайта и кликане върху бутона explore).
6. Отворете Web.config файлът в главната директория.
7. Потърсете XML тагът „connectionStrings”, в него намерете вложен таг <add/> с атрибут name=”RdeConnectionString”.
8. За да използвате желаната от Вас база данни, променете атрибутът connectionString с пътят към желаната от Вас база данни.
9. Рестартирайте приложението като от IIS кликнете с десен бутон върху сайта и изберете “restart”.

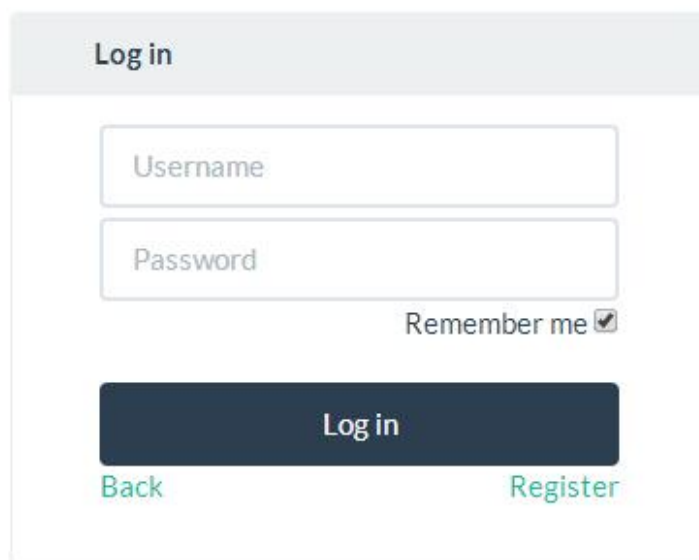
4.2 Използване на системата

След стартиране на системата пред вас ще се появи стартовият екран.



4.2.1 Автентикация

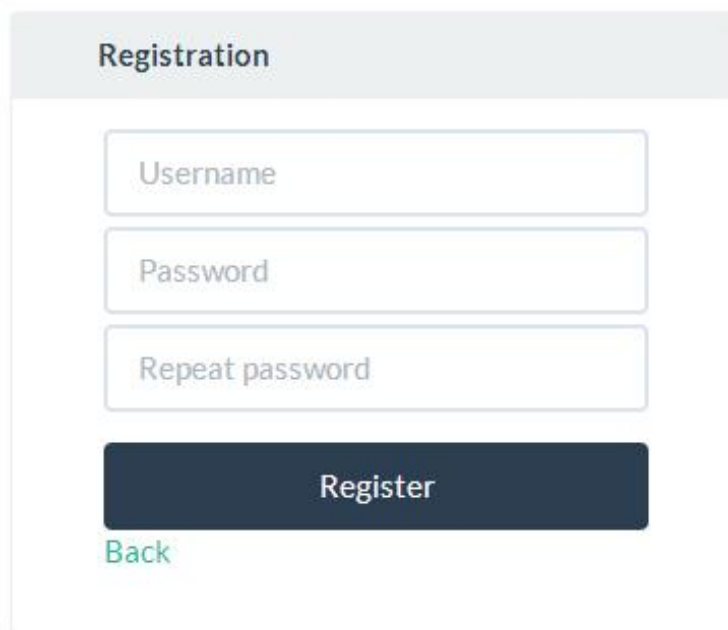
За вход в системата, моля кликнете върху бутона “Login” горе в дясно. Пред Вас ще се появи следният прозорец.



В случай че имате вече създадени потребителско име и парола, можете да ги въведете в текстовите кутии съответно “Username” и “Password”, и да кликнете върху бутонът Log in, за вход в системата. Кутията с отметка „Remember me” служи за запомняне на потребителската сесия, в резултат на което при повторно посещаване на

приложението вашите входно име и парола ще бъдат запазени и няма да има нужда от повторно въвеждане на входното име и паролата.

В случай че нямате регистриран потребител все още, моля кликнете върху бутона „Register”. Пред Вас ще излезе следният прозорец.

A screenshot of a web registration form. At the top, there is a light gray header bar with the word "Registration" in bold black text. Below this, the form is contained within a white box with a thin gray border. It features three stacked text input fields with light blue borders and placeholder text: "Username", "Password", and "Repeat password". Below these fields is a dark blue rectangular button with the word "Register" in white text. At the bottom left of the form, there is a green text link that says "Back".

В текстовата кутия “Username” можете да въведете желаното от Вас потребителско име. В текстовите кутии „Password” и “Repeat password” моля въведете една и съща последователност от символи, която ще служи за ваша парола, и която ще въвеждате заедно с потребителското си име при вход в системата. След като успешно попълните потребителското име и паролата, моля кликнете на бутон “Register”. Моля обърнете внимание, че при вече заето име, или несъответстваща парола, ще бъдете помолени да въведете на ново коректни потребителско име и парола.

4.2.2 Създаване на проекти

За създаване на проекти, трябва първо да бъдете регистриран в системата. В случай че не знаете как да се регистрирате, моля прочетете на ново точка 4.2.2. След вход в системата, кликнете върху бутон “Project” от стартовия прозорец. След това изберете “New Project”.

При натискане на бутона „New Project” пред вас ще излезе следният диалогов прозорец.

Create new project

The project name is too short. 5 symbols at least.

Project name

Visibility

Public Private

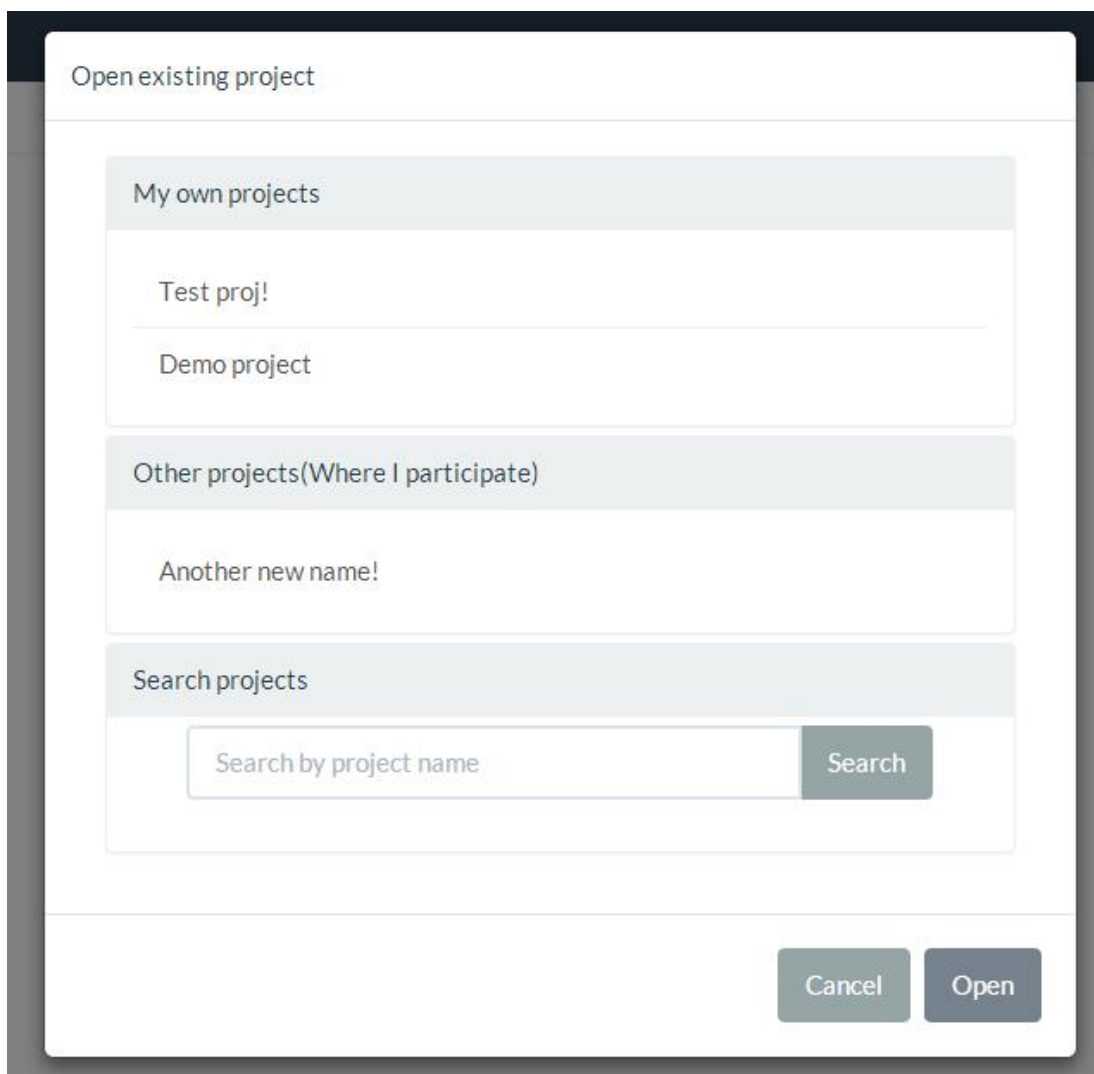
The project visibility determines who will be able to see your project. Public means that it is visible to anyone on the internet. Private means that only you, and the people who are part of the project can see it's contents.

Cancel Create

В текстовата кутия за име на проекта въведете желаното от Вас име. Групата бутони “Visibility” определят виимостта на проекта. Можете да избирате между два вида видимост – публична, и частна. Ако изберете проектът Ви да бъде публичен, то той може да бъде разглеждан и от потребители които не са влезли в системата. Ако изберете проектът да бъде частен, то той ще може да бъде разглеждан само от потребители, които Вие сте добавили в проекта. След избиране на име и на видимост на проекта, моля кликнете върху бутона “Create”.

4.2.3 Отваряне на проекти

За отваряне на вече създадени проекти, моля кликнете върху бутон “Project” от стартовият прозорец, и след това изберете „Open Project”. Пред Вас ще се появи следният диалогов прозорец.



Това е примерен прозорец със съществуващи проекти, Вашият най-вероятно ще изглежда по-друг начин. Диалоговият прозорец е разделен на 3 части. В секция “My own projects” ще намерите проекти, които Вие сте създали с текущият си потребител. В секция “Other projects(Where I participate)” ще намерите проекти в които сте добавен като участник. В секция Search projects можете да въведете името на даден проект за който сте чули, например от Ваш приятел програмист, и искате да разгледате. Моля обърнете внимание че след натискане на бутон “Search” показаните резултати ще бъдат само на

проекти които са с ПУБЛИЧНА видимост. В случай, че проекта е частен, и Вие не участвате в него, няма да можете да го разгледате.

4.2.4 Настройки на проектите

След създаване на проекта, неговия притежател може да сменя името му, видимостта му, и да добавя нови хора към проекта. За да видите диалоговия прозорец с настройките на проекта, кликнете върху бутона “Project”, а след това “Manage my projects”. Ще видите диалогов прозорец, подобен на следния.

The screenshot shows a settings dialog for a project named "Test proj!". At the top, there is a "Select project" dropdown, the project name "Test proj!" with a star icon, and a "Delete" button. Below this, the "Name" field contains "Test proj!". The "Visibility" section has two buttons: "Public" (selected) and "Private". The "Participants" section contains a table with the following data:

Name	Access level	Action
Keremidko	OWNER	
Tester	<div>E E/C E/C/D</div>	<div>Remove</div>

Below the table is a "Search users" section with a text input "Search by user name" and a "Search" button. At the bottom right, there are "Close" and "Save" buttons.

За промяна на името на проекта, сменете стойността на текстовата кутия “Name”. За промяна на видимостта на проекта – активирайте алтернативният “Visibility” бутон. За смяна на избраният проект – кликнете върху “Select project”, от там ще ви изкочи падащо меню, със другите проекти които притежавате. За изтриване на текущия проект – кликнете “Delete”. За да добавите участник към текущият проект – напишете името му във “Search by user name” текстовата кутия, и кликнете върху бутон “Search”. Отдолу ще ви излезе списък с всички регистрирани в системата потребители, които отговарят на зададените критерии за търсене. Търсенето не е чувствително към главни и малки букви, и работи ако въведеният текст е част от името на потребителя.

На долната картинка, виждате резултатът от търсенето на потребители, по въведен текст “te”. Валидните резултати са „Test” и “Tester”.

Name	Access level	Action
Test	<div><div>E</div><div>E/C</div><div>E/C/D</div></div>	<button>Add</button>
Tester	<div><div>E</div><div>E/C</div><div>E/C/D</div></div>	<button>Add</button>

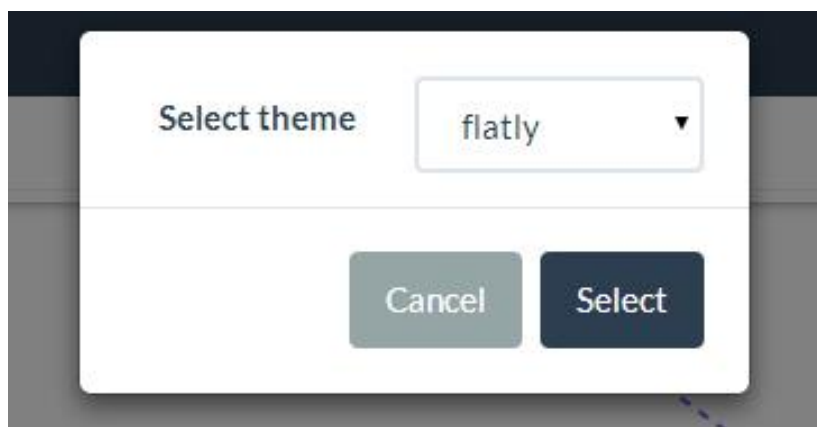
За да добавите потребител към Вашият проект, кликнете върху бутонът “Add”. Добавяйки даден потребител към проекта си, Вие му давате възможност да променя различните файловете които съществуват в проекта. За да разширите възможностите на потребителя, можете да смените нивото му на достъп, от групата с бутони “Access Level”.

- Ниво E - Edit Потребителят има прави само да прави промени по файловете.
- Ниво E/C – Edit/Create Потребителят има право освен промени по файловете, и да създава нови файлове
- Ниво E/C/D – Edit/Create/Delete Потребителят има право да променя съдържанието на файловете, да създава нови файлове, и да трие вече съществуващи файлове.

Когато приключите с правенето на промените, кликнете върху бутона “Save”

4.2.5 Смяна на изгледа на системата

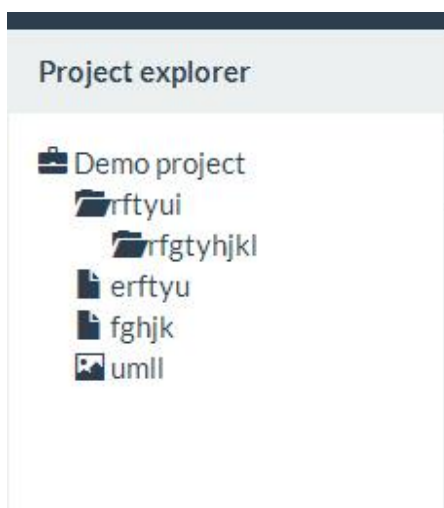
Системата поддържа 16 различни теми на потребителския интерфейс. В случай че темата по подразбиране не Ви допада, или имате чувствителност към светлината от монитора, можете да смените активната тема като кликнете на бутон “Options” и след това кликнете върху „Themes”. Диалогът който ще видите е подобен на този долу.



От падащият списък “Select theme” можете да изберете една от 16те теми. След това изберете Select. Информацията за това коя тема се използва се пази във Cookie (бисквитка), така че ако си смените browser-a, или решите да си изтриете cookie-тата на текущия browser, ще Ви бъде предоставена темата по подразбиране (Cerulean).

4.2.6 Работа с Project Explorer-a

Project Explorer-a е панелът в ляво на вашият екран. След отваряне на даден проект, в него се зареждат различните артикули на проекта. Примерна картинка с изгледа на project explorer-a можете да видите отдолу.



Името на отвореният проект стои най-отгоре (иконката му е куфарче). Под него в дървовидна структура са подредени всички файлове и папки. За отваряне на дадена папка или файл – кликнете два пъти с левият бутон на мишката върху нея. Ако сте кликнули върху отворена папка – то съдържанието и ще ви се покаже отдолу. Ако папката е била затворена – съдържанието и ще се скрие. Ако сте кликнули върху файл, то в дясно на Project Explorer-а ще ви се визуализира съдържанието на папката. За повече информация за работата с файлове – в точка 4.2.8.

За допълнителни опции при работа с файловете, папките и проекта – моля кликнете десен бутон върху едно от трите. Ще видите контекстното меню, на съответният артикул.

Контекстно меню проект

- Open – показва съдържанието на проекта
- Close – скрива съдържанието на проекта
- Add folder – добавя нова папка към структурата на проекта
- Add document – добавя нов документ към структурата на проекта
- Add diagram – Добавя нова диаграма към структурата на проекта

Контекстно меню папка

- Open – показва съдържанието на папката
- Close – скрива съдържанието на папката
- Add folder – добавя папка към кликнатата папка
- Add document – добавя документ към кликнатата папка
- Add diagram – добавя диаграма към кликнатата папка
- Delete – изтрива папката

Контекстно меню диаграма/документ

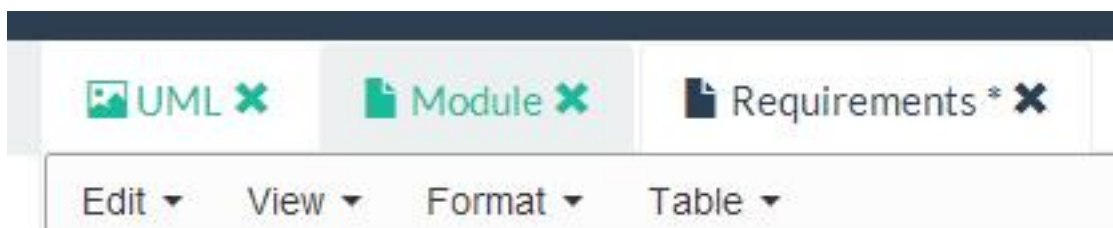
- Open – показва съдържанието на документа/диаграмата
- Delete – изтрива документа/диаграмата

Освен от Project Explorer-а, към проекта имате възможността да добавяте папки, диаграми и документи от менюто File.

- New folder – добавя нова папка
- New document – добавя нов документ
- New diagram – добавя нова диаграма
- Save – Изпраща съдържанието на отворената диаграма/документ за съхранение на сървъра.

4.2.7 Навигация през отворените диаграми/документи

След като отворите даден документ/диаграма, в работната област (в средата на монитора) ще видите под формата на табове да се добавят имената на отворените от вас документи. Отдолу виждате примерна картинка на три отворени артикула.



За да отворите даден артикул, кликнете с лявото копче на мишката един път върху името му. Ако артикулът е вече отворен – няма да видите никаква разлика с предишната. Моля обърнете внимание, че при кликане върху различните табове URL-ът на browser-а се сменя. URL-а отговаря точно на всеки един артикул.

Пример:

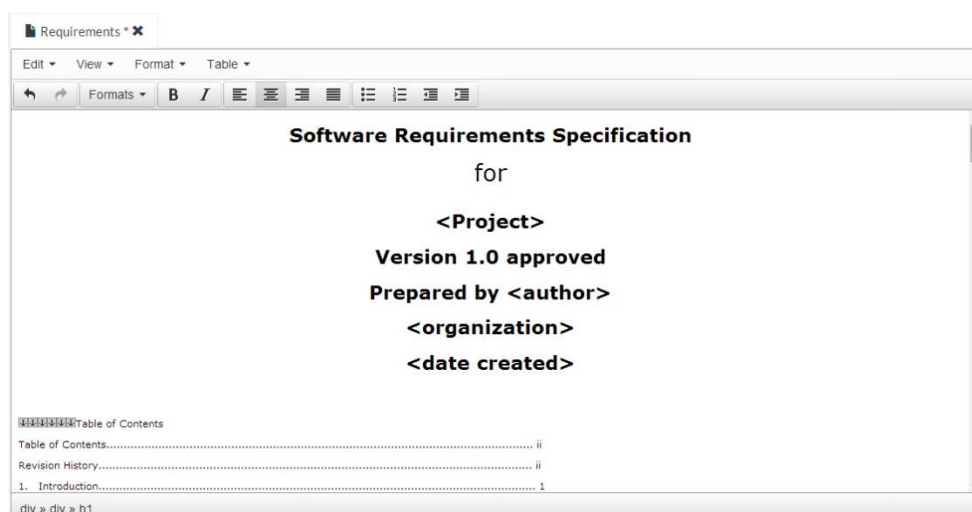
<http://localhost:58890/#/Project/51/Document/75>

<http://localhost:58890/#/Project/51/Diagram/72>

При желание да покажете отвореният от Вас артикул на някой ваш колега програмист просто копирайте линка на browser-а и му го дайте. Системата автоматично ще отведе вашият колега до посоченият от вас първоначално артикул.

4.2.8 Работа с документи

След създаването, и отваряне на документ, пред вас ще се появи следният текстов редактор :



Обърнете внимание че при създаване на текстов документ, в диалоговият прозорец излиза отметка “Use requirements template” при кликане върху отметката, и създаване на документа, ще получите документ със вече готова структура за вашите изисквания към проекта.

Интерфейсът на редакторът се състои от падащи менюта, и бутони. В падащо меню “Edit” ще намерите функции за редактиране на текстовото съдържание. Поддържат се :

- Undo – връщане 1 действие назад
- Redo – едно действие напред
- Cut – изрязване на текст
- Copy – копиране на текст
- Paste – поставяне на текст

От падащо меню “Format” са налични функции за форматиране на текста като :

- Bold - одепеляване на текста
- Italic – накланяне на текста
- Underline – подчертаване на текста
- Strikethrough – задраскване на текста
- други

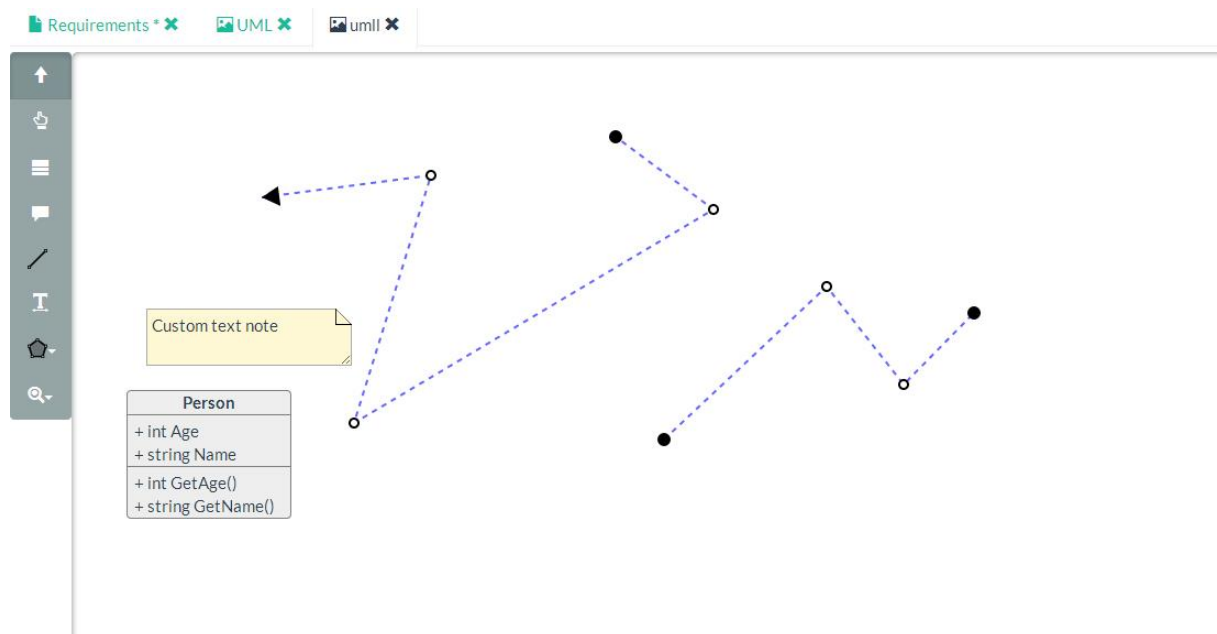
От падащо меню “Table” имате възможност за избиране на формата на таблицата която искате да създадете. Възможности за добавяне на ред, клетка, настройки на клетки и редове и други.

С бутоните под падащите менюта имате опции за :

- Форматиране на маркираният текст
- Подравняване на текста
- Създаване на списъчна структура
- Увеличаване/намаляване на отместването.

4.2.9 Работа с диаграми

Освен създаване на документи, системата предоставя и възможност за създаване на диаграми, целта на които е да покажат структурата и дизайна на проектираната софтуерна система. За създаване на диаграма кликнете върху менюто “File” и след това кликнете върху “New Diagram”. В работния прозорец ще се зареди едитор, подобен на показаният долу.



В средата виждате примерно запълване на работната зона на едитора. В ляво виждате инструментите които са налични за работа със едитора. Функционалността на всеки от инструментите от горе на долу е както следва :

- Стрелка преработване – това е стандартният инструмент с който можете да преработвате създадените в едитора обекти.
- Ръка преместване – с този инструмент можете да местите обектите в едитора като кликнете и преместите мишката.
- Създаване на клас диаграма – този инструмент създавате клас диаграма. Кликнете на произволно място в работното поле за да създадете диаграмата.
- Създаване на бележка – с този инструмент създавате текстови бележки. Кликнете на избрано от вас място в работното поле за създаване на бележка.
- Линия – с този инструмент създавате начипени линии. С всяко кликуване – добавяте нова точка към линията. За да прекратите създаването на текущата линия – или натиснете enter, или сменете инструмента.
- Текст – с този инструмент създавате текст. Кликнете някъде в едитора за да създадете текстово кутия, след това изберете инструмента за редактиране за да смените съдържанието на текста.
- Лупа – този инструмент има падащо меню с два под инструмента – лупа за увеличение, и лупа за намаляване. С лупата за увеличение Вие приближавате към работната зона, с лупата за отдалечаване – се отдалечавате от работната зона.

Глава 5

Заклучение, предложения за развитие и използване

Заклучение

С развитието на интернет технологиите се появяват все повече и повече възможности за web базираните приложения. Предлагащата от тях функционалност бива многоплатформена, достъпна от много точки. Те спестяват на потребителя нуждата от инсталиране и обновяване на приложението. При наличието на програмно задание, за което не са нужни големи изчислителни ресурси, интернет технологиите се явяват подходящо решение, заради горе изброените предимства.

В текущия документ се предлага система за изготвяне на изискванията и проектирането на софтуерно проекти, която спомага процеса на разработване на софтуерни продукти, и олеснява работата на разработчиците. Тя съдържа нововъведения и функционалности, които ще подобрят комуникацията между разработчиците, като по този начин ще намали възможните грешки които биха настъпили при софтуерната разработка.

Системата се характеризира с поддържането на различни типа артикули, които програмистите могат да създават – документи и диаграми. В документите се описват изискванията на системата, и различни бележки по имплементацията, в диаграмите се показва нагледно структурата която ще трябва да бъде имплементирана.

Осигурено е ниво на защита на проектите, чрез имплементирането на нива на видимост на проекта, също както и имплементирането на различни роли за отделните разработчици използващи системата.

Лесният и интуитивен интерфейс на системата, и добре направената документация и ръководство за ползване я правят лесна за усвояване от всеки един разработчик, без нужда от допълнително обучение.

Предложения за развитие и използване

- Създаване на покани за участие в проект. В момента разработчикът се добавя автоматично щом автора го намери.
- Създаване на допълнителни форми за рисуване във диаграмният редактор.

- Подобряване на външния вид на някои от обектите спрямо настоящите UML стандарти в диаграмният редактор.
- Добавяне на четка към наличните инструменти в диаграмният редактор. Целта на четката е по-лесно изобразяване с помощта на мишката на някои сложни форми от страна на разработчиците.
- Добавяне на Cut/Copy/Paste опции за диаграмите и документите на проекта.
- Добавяне на Undo/Redo функционалности за диаграмният редактор.

Изходен код на системата

```
namespace Requirements_and_Design_environment
{
    public class BundleConfig
    {
        private const string LIBRARIES_PATH = "~/JS_Libraries/";

        private const string SCRIPTS_PATH = "~/Scripts/";

        private const string STYLES_PATH = "~/Styles/";

        // For more information on bundling, visit http://go.microsoft.com/fwlink/?LinkId=301862
        public static void RegisterBundles(BundleCollection bundles)
        {
            RegisterLibraries(bundles);
            RegisterClientScripts(bundles);
            RegisterStyles(bundles);
        }

        private static void RegisterLibraries(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/angular").Include(
                LIBRARIES_PATH + "angular.js",
                LIBRARIES_PATH + "angular-cookies.js",
                LIBRARIES_PATH + "angular-ui-router.js",
                LIBRARIES_PATH + "angular-animate.js",
                LIBRARIES_PATH + "ui-tinymce.js",
                LIBRARIES_PATH + "ui-bootstrap-0.11.0.js"));

            bundles.Add(new ScriptBundle("~/bundles/tinymce").Include(
                LIBRARIES_PATH + "tinymce/js/tinymce/tinymce.min.js"));

            bundles.Add(new ScriptBundle("~/bundles/jquery").Include((
                LIBRARIES_PATH + "jquery-1.10.2.js")));
        }

        private static void RegisterClientScripts(BundleCollection bundles)
        {

```

```
//App bundle - bundle-а със всички модули за приложението
bundles.Add(new ScriptBundle("~/bundles/app").Include(
    SCRIPTS_PATH + "app.js"));

bundles.Add(new ScriptBundle("~/bundles/app/controllers").Include(
    SCRIPTS_PATH + "/controllers/MenuController.js",
    SCRIPTS_PATH + "/controllers/AccountController.js",
    SCRIPTS_PATH + "/controllers/ProjectExplorerController.js",
    SCRIPTS_PATH + "/controllers/BaseController.js",
    SCRIPTS_PATH + "/controllers/WorkingAreaController.js",
    SCRIPTS_PATH + "/controllers/ToolboxController.js",
    SCRIPTS_PATH + "/controllers/DiagramController.js",
    SCRIPTS_PATH + "/controllers/modals/NewProjectController.js",
    SCRIPTS_PATH + "/controllers/modals/OpenProjectController.js",
    SCRIPTS_PATH + "/controllers/modals/ManageProjectsController.js",
    SCRIPTS_PATH + "/controllers/modals/SelectThemeController.js",
    SCRIPTS_PATH + "/controllers/modals/NewItemController.js",
    SCRIPTS_PATH + "/controllers/modals/SaveItemController.js"));

bundles.Add(new ScriptBundle("~/bundles/app/services").Include(
    SCRIPTS_PATH + "/services/ModalService.js",
    SCRIPTS_PATH + "/services/AuthorizationService.js",
    SCRIPTS_PATH + "/services/DeserializationService.js",
    SCRIPTS_PATH + "/services/ProjectService.js",
    SCRIPTS_PATH + "/services/ToolsService.js",
    SCRIPTS_PATH + "/services/ErrorHandlerService.js"
));

bundles.Add(new ScriptBundle("~/bundles/app/models").Include(
    SCRIPTS_PATH + "/models/Item.js"));

bundles.Add(new ScriptBundle("~/bundles/app/filters").Include(
    SCRIPTS_PATH + "/filters/ProjectFilters.js"
));

bundles.Add(new ScriptBundle("~/bundles/app/directives").Include(
    SCRIPTS_PATH + "/directives/ContextMenu.js",
    SCRIPTS_PATH + "/directives/Resizer.js",
    SCRIPTS_PATH + "/directives/StopPropagation.js",
    SCRIPTS_PATH + "/directives/DiagramView.js",
    SCRIPTS_PATH + "/directives/Contenteditable.js",
    SCRIPTS_PATH + "/directives/SvgTextInput.js",
    SCRIPTS_PATH + "/directives/Draggable.js",
    SCRIPTS_PATH + "/directives/Line/Line.js",
    SCRIPTS_PATH + "/directives/Text/Text.js",
    SCRIPTS_PATH + "/directives/Ellipse/Ellipse.js",
    SCRIPTS_PATH + "/directives/vector_drag/VectorDrag.js",
    SCRIPTS_PATH + "/directives/Rectangle/Rectangle.js",
    SCRIPTS_PATH + "/directives/ClassDiagram/ClassDiagram.js",
    SCRIPTS_PATH + "/directives/Comment/Comment.js",
    SCRIPTS_PATH + "/directives/SvgWrapper/SvgWrapper.js"));
}

private static void RegisterStyles(BundleCollection bundles)
{
    bundles.Add(new StyleBundle("~/Content/css").Include(
        STYLES_PATH + "Site.css"));
}
```

```
    }  
}  
  
namespace Requirements_and_Design_environment  
{  
    public class FilterConfig  
    {  
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)  
        {  
            filters.Add(new HandleErrorAttribute());  
        }  
    }  
}  
  
namespace Requirements_and_Design_environment  
{  
    public class RouteConfig  
    {  
        public static void RegisterRoutes(RouteCollection routes)  
        {  
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");  
  
            routes.MapRoute(  
                name: "Template",  
                url: "Template/Get/{name}",  
                defaults: new { action = "Get", controller = "Template", name =  
UrlParameter.Optional }  
            );  
  
            routes.MapRoute(  
                name: "Modal",  
                url: "Template/Modal/{name}",  
                defaults: new { action = "Modal", controller = "Template", name =  
UrlParameter.Optional }  
            );  
  
            routes.MapRoute(  
                name: "Default",  
                url: "{controller}/{action}/{id}",  
                defaults: new { action = "Index", controller = "Home", id =  
UrlParameter.Optional }  
            );  
        }  
    }  
}  
  
namespace Requirements_and_Design_environment  
{  
    public static class WebApiConfig  
    {  
        public static void Register(HttpConfiguration config)  
        {  
            // Web API configuration and services  
  
            // Web API routes
```



```
        config.MapHttpAttributeRoutes();

        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{action}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );
    }
}

namespace Requirements_and_Design_environment.Controllers
{
    [Authorize]
    public class AccountController : ApiController
    {
        private IUsersRepository usersRepository { get; set; }

        public AccountController()
        {
            usersRepository = new UsersRepository();
        }

        [HttpPost]
        [AllowAnonymous]
        [System.Web.Mvc.ValidateAntiForgeryToken]
        public IHttpActionResult Login(LoginModel model)
        {
            if (ModelState.IsValid && WebSecurity.Login(model.UserName, model.Password,
persistCookie: model.RememberMe))
            {
                return Ok();
            }

            return BadRequest("The user name or password provided is incorrect.");
        }

        [HttpPost]
        [AllowAnonymous]
        [System.Web.Mvc.ValidateAntiForgeryToken]
        public IHttpActionResult Register(RegisterModel model)
        {
            string message = "";
            if (ModelState.IsValid)
            {
                // Attempt to register the user

                try
                {
                    WebSecurity.CreateUserAndAccount(model.UserName, model.Password);
                    WebSecurity.Login(model.UserName, model.Password);
                    return Ok();
                }
                catch (MembershipCreateUserException e)
                {

```

```
        message = ErrorCodeToString(e.StatusCode);
    }
    catch (Exception e)
    {
        System.Diagnostics.Debug.WriteLine(e.ToString());
    }
}
else {
    message = "Incorrect input data. User name is mandatory, password should be
atleast 6 symbols, and both password should match.";
}

// If we got this far, something failed, redisplay form
return BadRequest(message);
}

[HttpGet]
[AllowAnonymous]
public bool IsUserLogged()
{
    return WebSecurity.IsAuthenticated;
}

[HttpGet]
public object GetProfile()
{
    UserProfile user = usersRepository.GetCurrentUserProfile();
    var projects = from p in user.Participations
        //Избира всички проекти в които участва user-a
        select new
        {
            ID = p.ProjectID,
            AccessLevel = p.AccessLevel,
            Name = p.ProjectReference.Name,
            Visibility = p.ProjectReference.Visibility,
            //избира всички user-и които участват в проекта на user-a
            Participants = from par in p.ProjectReference.Participations
                select new
                {
                    UserName = par.UserReference.UserName,
                    AccessLevel = par.AccessLevel,
                }
        };
    var viewModel = new
    {
        UserName = user.UserName,
        Projects = projects
    };
    return viewModel;
}

[HttpGet]
public string[] GetUsersByName(string name)
{
    return usersRepository.GetUsersByName(name);
}

[HttpPost]
```

```
public IHttpActionResult Logout()
{
    WebSecurity.Logout();
    return Ok();
}

private static string ErrorCodeToString(MembershipCreateStatus createStatus)
{
    // See http://go.microsoft.com/fwlink/?LinkID=177550 for
    // a full list of status codes.
    switch (createStatus)
    {
        case MembershipCreateStatus.DuplicateUserName:
            return "User name already exists. Please enter a different user name.";

        case MembershipCreateStatus.DuplicateEmail:
            return "A user name for that e-mail address already exists. Please enter a
different e-mail address.";

        case MembershipCreateStatus.InvalidPassword:
            return "The password provided is invalid. Please enter a valid password
value.";

        case MembershipCreateStatus.InvalidEmail:
            return "The e-mail address provided is invalid. Please check the value and
try again.";

        case MembershipCreateStatus.InvalidAnswer:
            return "The password retrieval answer provided is invalid. Please check the
value and try again.";

        case MembershipCreateStatus.InvalidQuestion:
            return "The password retrieval question provided is invalid. Please check
the value and try again.";

        case MembershipCreateStatus.InvalidUserName:
            return "The user name provided is invalid. Please check the value and try
again.";

        case MembershipCreateStatus.ProviderError:
            return "The authentication provider returned an error. Please verify your
entry and try again. If the problem persists, please contact your system administrator.";

        case MembershipCreateStatus.UserRejected:
            return "The user creation request has been canceled. Please verify your
entry and try again. If the problem persists, please contact your system administrator.";

        default:
            return "An unknown error occurred. Please verify your entry and try again.
If the problem persists, please contact your system administrator.";
    }
}

namespace Requirements_and_Design_environment.Controllers
{
    public class HomeController : Controller
```

```
{
    //
    // GET: /Default/
    public ActionResult Index()
    {
        string theme;
        if(Request.Cookies["theme"] != null)
            theme = Request.Cookies["theme"].Value;
        else
            theme = "cerulean";
        ViewData["theme"] = theme;
        return View();
    }
}

namespace Requirements_and_Design_environment.Controllers
{
    [Authorize]
    public class ItemController : ApiController
    {
        private IProjectRepository projectRepository { get; set; }
        private IUsersRepository usersRepository { get; set; }

        public ItemController()
        {
            projectRepository = new ProjectRepository();
            usersRepository = new UsersRepository();
        }

        //TODO : Да се провери дали user-а има create права в текущия проект
        //TODO : Да се провери дали parentId-то на item-а е от тип folder
        //TODO : Уникално име на Item-а
        public IHttpActionResult CreateItem(Item item, int projectId, int templateId)
        {
            var user = System.Web.HttpContext.Current.User.Identity.Name;
            var proj = projectRepository.GetProject(projectId);

            var participation = proj.Participations.Find(x => x.UserReference.UserName == user);
            if (participation != null
                && participation.AccessLevel >= Accessibility.Read_Edit_Create
                || participation.AccessLevel == Accessibility.Owner)
            {
                try
                {
                    {
                        item = projectRepository.AddItemToProject(item, projectId, templateId);
                        return Ok<Item>(item);
                    }
                }
                catch (Exception e)
                {
                    {
                        return BadRequest();
                    }
                }
            }
            else
            {
                return BadRequest("You don't have rights to create items. Please ask your project manager to provide you with the rights before proceeding again.");
            }

            //TODO : Да се провери дали user-а има Edit права в текущия проект
            [HttpPost]
        }
    }
}
```

```
public IHttpActionResult UpdateItem(ItemUpdateModel model)
{
    try
    {
        projectRepository.UpdateItem(model.ItemId, model.Content);
        return Ok();
    }
    catch (Exception e)
    {
        return BadRequest();
    }
}

[HttpDelete]
//TODO : Да се провери дали user-а има Delete права в проекта
public IHttpActionResult DeleteItem(int itemId)
{
    var item = projectRepository.GetItem(itemId);
    var user = System.Web.HttpContext.Current.User.Identity.Name;
    var proj = item.Project;

    var participation = proj.Participations.Find(x => x.UserReference.UserName == user);
    if (participation != null
        && participation.AccessLevel >= Accessibility.Read_Edit_Create_Delete
        || participation.AccessLevel == Accessibility.Owner)
    {
        try
        {
            projectRepository.DeleteItem(itemId);
            return Ok();
        }
        catch (Exception e)
        {
            return BadRequest();
        }
    }
    else
    {
        return BadRequest("You don't have rights to delete this item. Please ask your
project manager to provide you with the rights before proceeding again.");
    }
}

namespace Requirements_and_Design_environment.Controllers
{
    [Authorize]
    public class ProjectController : ApiController
    {
        private IProjectRepository projectRepository { get; set; }
        private IUsersRepository usersRepository { get; set; }

        public ProjectController()
        {
            projectRepository = new ProjectRepository();
            usersRepository = new UsersRepository();
        }

        [HttpGet]
        [AllowAnonymous]
        //TODO : Да се направи проверка за видимостта на проекта
    }
}
```

```
public object OpenProject(int id)
{
    var proj = projectRepository.GetProject(id);
    proj.Items.Sort();
    return new
    {
        ID = proj.ID,
        Name = proj.Name,
        Items = proj.Items
    };
}

[HttpGet]
[AllowAnonymous]
public object[] GetPublicProjectsByName(string name)
{
    var projects = projectRepository.GetProjectsByName(name);
    var result = (from p in projects
        where p.Visibility == ProjectVisibility.Public
        select new {
            Name = p.Name,
            ID = p.ID
        }).ToArray();
    return result;
}

[HttpPost]
public IHttpActionResult CreateProject(Project model, [FromUri]ProjectVisibility
visibility)
{
    try
    {
        model.Visibility = visibility;
        var profile = usersRepository.GetCurrentUserProfile();
        usersRepository.Dispose();
        projectRepository.CreateProject(model, profile);
        return Ok();
    }
    catch (DbUpdateException e)
    {
        return BadRequest("A project with this name already exists.");
    }
    catch (Exception e)
    {
        return BadRequest("An error occurred while trying to create a new project.");
    }
}

//TODO: Да се добави проверка дали текущия user е собственик на проекта
//TODO: Логиката да се премести в repository-то
[HttpPost]
public IHttpActionResult UpdateProject([FromBody]ProjectSaveModel model, [FromUri]string
name)
{
    try {
        Project proj = projectRepository.GetProject(name);
        proj.Name = model.Name;
        proj.Visibility = model.Visibility;
    }
```

```
proj.RemoveAllParticipantsButOwner();
List<KeyValuePair<UserProfile, Accessibility>> users = new
List<KeyValuePair<UserProfile, Accessibility>>();

for (int i = 0; i < model.Participants.Length; i++)
{
    if (model.Participants[i].AccessLevel == Accessibility.Owner)
        continue;
    var user = usersRepository.GetProfileByName(model.Participants[i].UserName);
    users.Add(new KeyValuePair<UserProfile, Accessibility>(user,
model.Participants[i].AccessLevel));
}
usersRepository.Dispose();
for (int i = 0; i < users.Count; i++)
{
    proj.Participations.Add(new Participation
    {
        ProjectReference = proj,
        UserReference = users[i].Key,
        AccessLevel = users[i].Value
    });
    projectRepository.GetContext().Entry(users[i].Key).State =
EntityState.Unchanged;
}

    projectRepository.GetContext().SaveChanges();
    return Ok();
}
catch (Exception e)
{
    return BadRequest();
}
}

//TODO : Да се провери дали е owner текущия user
//TODO : да се добави Confirm window на клиентската част
[HttpDelete]
public IHttpActionResult DeleteProject(string name)
{
    try
    {
        projectRepository.DeleteProject(name);
        return Ok();
    }
    catch (Exception e)
    {
        return BadRequest();
    }
}
}

namespace Requirements_and_Design_environment.Controllers
{
    //TODO: Да се сложи еTag за кеширане на темплейтите
    public class TemplateController : Controller
    {
        [HttpGet]
```

```
        public PartialViewResult Get(string name)
        {
            return PartialView(String.Format("~/Views/Templates/{0}.cshtml", name));
        }

        [HttpGet]
        public PartialViewResult Modal(string name)
        {
            return PartialView(String.Format("~/Views/Templates/Modals/{0}.cshtml", name));
        }
    }
}

public sealed class ExceptionUtility
{
    // All methods are static, so this can be private
    private ExceptionUtility()
    { }

    // Log an Exception
    public static void LogException(Exception exc, string source)
    {
        // Include enterprise logic for logging exceptions
        // Get the absolute path to the log file
        string logFile = "App_Data/ErrorLog.txt";
        logFile = HttpContext.Current.Server.MapPath(logFile);

        // Open the log file for append and write the log
        StreamWriter sw = new StreamWriter(logFile, true);
        sw.WriteLine("***** {0} *****", DateTime.Now);
        if (exc.InnerException != null)
        {
            sw.Write("Inner Exception Type: ");
            sw.WriteLine(exc.InnerException.GetType().ToString());
            sw.Write("Inner Exception: ");
            sw.WriteLine(exc.InnerException.Message);
            sw.Write("Inner Source: ");
            sw.WriteLine(exc.InnerException.Source);
            if (exc.InnerException.StackTrace != null)
            {
                sw.WriteLine("Inner Stack Trace: ");
                sw.WriteLine(exc.InnerException.StackTrace);
            }
        }
        sw.Write("Exception Type: ");
        sw.WriteLine(exc.GetType().ToString());
        sw.WriteLine("Exception: " + exc.Message);
        sw.WriteLine("Source: " + source);
        sw.WriteLine("Stack Trace: ");
        if (exc.StackTrace != null)
        {
            sw.WriteLine(exc.StackTrace);
            sw.WriteLine();
        }
        sw.Close();
    }
}
```



```
namespace Requirements_and_Design_environment.Infrastructure
{
    public class MembershipInitializer
    {
        public static void InitializeLoginSystem()
        {
            Database.SetInitializer<RdeDbContext>(null);

            try
            {
                using (var context = new RdeDbContext())
                {
                    if (!context.Database.Exists())
                    {
                        // Create the SimpleMembership database without Entity Framework
migration schema
                        ((IObjectContextAdapter)context).ObjectContext.CreateDatabase();
                    }
                }
                WebSecurity.InitializeDatabaseConnection("RdeConnectionString", "UserProfile",
"UserId", "UserName", autoCreateTables: true);
            }
            catch (Exception ex)
            {
                throw new InvalidOperationException("The ASP.NET Simple Membership database
could not be initialized. For more information, please see
http://go.microsoft.com/fwlink/?LinkId=256588", ex);
            }
        }
    }
}

namespace Requirements_and_Design_environment.Infrastructure
{
    public partial class RdeDbContext : DbContext
    {
        public RdeDbContext()
            : base("name=RdeConnectionString")
        {
        }

        public DbSet<UserProfile> UserProfiles { get; set; }
        public DbSet<Project> Projects { get; set; }
        public DbSet<Participation> Participations { get; set; }
        public DbSet<Item> Items { get; set; }
        public DbSet<Template> Templates { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            //modelBuilder.Entity<Project>()
            //    .Property(e => e.Visibility)
            //    .IsFixedLength();
        }
    }
}

namespace Requirements_and_Design_environment.Models.ClientModels
```

```
{
    public class ItemUpdateModel
    {
        public int ItemId { get; set; }
        public string Content { get; set; }
    }
}

namespace Requirements_and_Design_environment.Models.ClientModels
{
    public class ProjectSaveModel
    {
        public string Name { get; set; }
        public ProjectVisibility Visibility { get; set; }
        public UserParticipantModel[] Participants { get; set; }
    }

    public class UserParticipantModel
    {
        public string UserName { get; set; }
        public Accessibility AccessLevel { get; set; }
    }
}

namespace Requirements_and_Design_environment.Models.Entities
{
    [Table("UserProfile")]
    [DataContract(IsReference=true)]
    public class UserProfile
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int UserId { get; set; }

        [DataMember]
        public string UserName { get; set; }

        [DataMember]
        public virtual List<Participation> Participations { get; set; }
    }

    public class LocalPasswordModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.",
MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }
    }
}
```

```
[DataType(DataType.Password)]
[Display(Name = "Confirm new password")]
[Compare("NewPassword", ErrorMessage = "The new password and confirmation password do
not match.")]
public string ConfirmPassword { get; set; }
}

public class LoginModel
{
    [Required]
    [Display(Name = "User name")]
    public string UserName { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [Display(Name = "Remember me?")]
    public bool RememberMe { get; set; }
}

public class RegisterModel
{
    [Required]
    [Display(Name = "User name")]
    public string UserName { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.",
MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not
match.")]
    public string ConfirmPassword { get; set; }
}

}
namespace Requirements_and_Design_environment.Models.Entities
{
    [Table("Items")]
    [DataContract]
    public class Item : IComparable<Item>
    {
        [Key]
        [DataMember]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int ID { get; set; }

        [DataMember]
        public int ParentID { get; set; }

        [DataMember]
```

```
        public int ProjectID { get; set; }

        [StringLength(50)]
        [DataMember]
        public string Name { get; set; }

        [DataMember]
        public ItemType Type { get; set; }

        [DataMember]
        public string Contents { get; set; }

        [IgnoreDataMember]
        [ForeignKey("ProjectID")]
        public virtual Project Project { get; set; }

        public int CompareTo(Item other)
        {
            return Type.CompareTo(other.Type);
        }
    }
}

namespace Requirements_and_Design_environment.Models.Entities
{
    [Table("Participations")]
    [DataContract(IsReference = true)]
    public class Participation
    {
        [ForeignKey("ProjectID")]
        [DataMember]
        public virtual Project ProjectReference { get; set; }
        [ForeignKey("UserID")]
        [DataMember]
        public virtual UserProfile UserReference { get; set; }

        [Key, Column(Order = 0)]
        public int ProjectID { get; set; }
        [Key, Column(Order = 1)]
        public int UserID { get; set; }
        [DataMember]
        public Accessibility AccessLevel { get; set; }
    }
}

namespace Requirements_and_Design_environment.Models.Entities
{
    [Table("Project")]
    [DataContract(IsReference=true)]
    public class Project
    {
        public Project()
        {
            Participations = new List<Participation>();
        }
    }
}
```

```
    }

    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int ID { get; set; }

    [StringLength(225)]
    [Index(IsUnique=true)]
    [DataMember]
    public string Name { get; set; }

    [DataMember]
    public ProjectVisibility Visibility { get; set; }

    [DataMember]
    public virtual List<Participation> Participations { get; set; }

    [DataMember]
    public virtual List<Item> Items { get; set; }

    public void RemoveAllParticipantsButOwner()
    {
        for (int i = 0; i < Participations.Count; i++)
        {
            if (Participations[i].AccessLevel == Accessibility.Owner)
                continue;
            Participations.Remove(Participations[i]);
            i--;
        }
    }
}

namespace Requirements_and_Design_environment.Models.Entities
{
    [Table("Templates")]
    public class Template
    {
        public int ID { get; set; }
        public string Contents { get; set; }
    }
}

namespace Requirements_and_Design_environment.Models.Enums
{
    public enum Accessibility
    {
        Owner = 1,
        Read_Edit = 2,
        Read_Edit_Create = 3,
        Read_Edit_Create_Delete = 4
    }
}
```

```
namespace Requirements_and_Design_environment.Models.Enums
{
    public enum ItemType
    {
        Folder = 1,
        Document = 2,
        Diagram = 3
    }
}

namespace Requirements_and_Design_environment.Models.Enums
{
    public enum ProjectVisibility
    {
        Public = 1,
        Private = 2
    }
}

namespace Requirements_and_Design_environment.Models.Interfaces
{
    public interface IProjectRepository : IDisposable
    {
        void DeleteProject(string name);

        void UpdateItem(int itemId, string content);

        void DeleteItem(int itemId);

        void CreateProject(Project model, UserProfile owner);

        Project[] GetProjectsByName(string name);

        Item GetItem(int id);

        Item AddItemToProject(Item item, int projectId, int templateId);

        Project GetProject(int id);
        Project GetProject(string name);
        DbContext GetContext();
    }
}

namespace Requirements_and_Design_environment.Models.Interfaces
{
    public interface IUsersRepository : IDisposable
    {
        UserProfile GetCurrentUserProfile();
        UserProfile GetProfileByName(string name);
        string[] GetUsersByName(string name);
    }
}

namespace Requirements_and_Design_environment.Models.Repositories
{
    public class BaseRepository : IDisposable
    {

```

```
#region " Declarations "

    private bool disposed = false;

    protected RdeDbContext dbContext = new RdeDbContext();

#endregion

#region " Dispose "

    protected virtual void Dispose(bool disposing)
    {
        if (!this.disposed)
        {
            if (disposing)
            {
                dbContext.Dispose();
            }
            this.disposed = true;
        }
    }

    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

#endregion

}

namespace Requirements_and_Design_environment.Models.Repositories
{
    public class ProjectRepository : BaseRepository, IProjectRepository
    {
        public void DeleteProject(string name)
        {
            var proj = dbContext.Projects.Where(x => x.Name == name).First();
            for (int i = 0; i < proj.Participations.Count; i++)
                dbContext.Participations.Remove(proj.Participations[i]);

            dbContext.Projects.Remove(proj);
            dbContext.SaveChanges();
        }

        public void DeleteItem(int itemId)
        {
            var item = dbContext.Items.Find(itemId);
            dbContext.Items.Remove(item);
            dbContext.SaveChanges();
        }

        public Item GetItem(int id) {
            return dbContext.Items.Find(id);
        }
    }
}
```

```
public void UpdateItem(int itemId, string content)
{
    dbContext.Items.Find(itemId).Contents = content;
    dbContext.SaveChanges();
}

public Project[] GetProjectsByName(string name)
{
    name = name.ToLower();
    var projects = from p in dbContext.Projects
                    where p.Name.ToLower().Contains(name)
                    select p;
    return projects.ToArray();
}

public Item AddItemToProject(Item item, int projectId, int templateId)
{
    if (templateId > -1)
        item.Contents = dbContext.Templates.Where(x => x.ID ==
templateId).First().Contents;
    var project = dbContext.Projects.Find(projectId);
    item.Project = project;
    dbContext.Items.Add((item));
    dbContext.SaveChanges();
    return item;
}

public void CreateProject(Project model, UserProfile owner)
{
    var participation = new Participation { ProjectReference = model, UserReference =
owner, AccessLevel = Accessibility.Owner };

    dbContext.Entry(model).State = EntityState.Added;
    dbContext.Entry(participation).State = EntityState.Added;
    dbContext.Entry(owner).State = EntityState.Unchanged;

    dbContext.SaveChanges();
}

public Project GetProject(int id)
{
    return dbContext.Projects.Find(id);
}

public Project GetProject(string name)
{
    return dbContext.Projects.Where(x => x.Name == name).First();
}

public DbContext GetContext()
{
    return dbContext;
}
}

namespace Requirements_and_Design_environment.Models.Repositories
{
```



```
public class UsersRepository : BaseRepository, IUsersRepository
{
    public UserProfile GetCurrentUserProfile()
    {
        return dbContext.UserProfiles.Where(x => x.UserName ==
HttpContext.Current.User.Identity.Name).FirstOrDefault();
    }

    public UserProfile GetProfileByName(string name)
    {
        return dbContext.UserProfiles.Where(x => x.UserName == name).FirstOrDefault();
    }

    public string[] GetUsersByName(string name)
    {
        name = name.ToLower();
        var users = dbContext.UserProfiles.ToList().FindAll(x =>
x.UserName.ToLower().Contains(name));
        return users.Select(x => x.UserName).ToArray<string>();
    }
}

RDE.controller("ManageProjectsController", function ($scope, $modalInstance, $http, $filter,
AuthorizationService) {

    $scope.userOwnedProjects =
$filter("UserOwnedProjects")(AuthorizationService.getProfile().Projects);
    $scope.selected = $scope.userOwnedProjects[0];
    $scope.selectedIndex = 0;
    $scope.dirtyModel = angular.copy($scope.selected);
    $scope.searchResults = [];
    $scope.unfilteredResults = [];

    $scope.showAlert = false;
    $scope.alertCallback = null;

    $scope.setSelected = function (proj, index) {
        var select = function () {
            $scope.searchResults = [];
            $scope.selected = proj;
            $scope.selectedIndex = index;
            $scope.dirtyModel = angular.copy(proj);
            $scope.dirtyModel.isDirty = false;
        };

        if ($scope.dirtyModel.isDirty) {
            $scope.showAlert = true;
            $scope.alertCallback = select;
        }
        else {
            select();
        }
    };

    $scope.setDirty = function () {
        $scope.dirtyModel.isDirty = true;
    };
};
```

```
$scope.save = function () {
    $http({
        url: '/api/Project/UpdateProject',
        method: 'POST',
        params: {
            name: $scope.selected.Name
        },
        data: $scope.dirtyModel
    }).success(function (data) {
        $scope.selected = $scope.dirtyModel;
        $scope.userOwnedProjects[$scope.selectedIndex] = $scope.dirtyModel;
        $scope.dirtyModel.isDirty = false;
    })
    .error(function (data) {
        console.log(data);
    });
};

$scope.close = function () {
    if ($scope.dirtyModel && $scope.dirtyModel.isDirty)
    {
        $scope.showAlert = true;
        $scope.alertCallback = $modalInstance.close;
    } else {
        $modalInstance.close();
    }
};

$scope.alertClick = function (useCallback) {
    if (useCallback)
        $scope.save($scope.dirtyModel);
    $scope.alertCallback();
    $scope.showAlert = false;
};

$scope.findUsers = function (name) {
    $scope.searchResults = [];
    $http.get('/api/Account/GetUsersByName?name='+name)
    .success(function (res) {
        $scope.unfilteredResults = res;
        $scope.searchResults = $filter("UsersNotInProject")($scope.dirtyModel, res);
    });
};

$scope.removeUser = function (index) {
    $scope.dirtyModel.Participants.splice(index,1);
    $scope.setDirty();
};

$scope.addUser = function (user) {
    $scope.dirtyModel.Participants.push(user);
    $scope.searchResults = $filter("UsersNotInProject")($scope.dirtyModel,
$scope.unfilteredResults);
    $scope.setDirty();
};
```

```
$scope.deleteProject = function ()
{
    $http.delete('/api/Project/DeleteProject', {
        params: {
            name : $scope.selected.Name
        }
    })
    .success(function () {
        $scope.userOwnedProjects.splice($scope.selectedIndex, 1);
        $scope.selected = $scope.userOwnedProjects[0];
        $scope.selectedIndex = 0;
        AuthorizationService.syncUserData();
    });
}
});

RDE.controller("NewItemController", function ($scope, $modalInstance, $cookies, type) {
    $scope.model = {
        Name: null,
        UseTemplate : false
    };

    $scope.type = type;

    $scope.create = function () {
        $modalInstance.close($scope.model);
    };

    $scope.cancel = function () {
        $modalInstance.dismiss('cancel');
    };
});

RDE.controller("NewProjectController", function ($scope, $modalInstance, $http,
AuthorizationService) {
    $scope.model = {
        ID : -1,
        Name: "",
        Visibility: 1,
        Participations : null
    }

    $scope.severError = false;
    $scope.serverMessage = "";

    $scope.nameMinLength = 5;

    $scope.create = function () {
        $http.post(
            '/api/Project/CreateProject?visibility='+$scope.model.Visibility,
            $scope.model
        ).success(function () {
            AuthorizationService.syncUserData();
            $modalInstance.close();
        }).error(function (data) {
            $scope.severError = true;
            $scope.serverMessage = data.Message;
        });
    };
});
```

```
    }

    $scope.closeAlert = function ()
    {
        $scope.serverError = false;
    }

    $scope.cancel = function () {
        $modalInstance.close();
    }
});

RDE.controller("OpenProjectController", function ($scope, $modalInstance, $http, ProjectService,
AuthorizationService) {
    $scope.projects = AuthorizationService.getProfile().Projects;

    $scope.selected = null;

    $scope.searchName = "";

    $scope.searchResults = [];

    $scope.open = function () {
        var request = ProjectService.loadProject($scope.selected.ID);
        request.success(function () {
            $modalInstance.close();
        });
    };

    $scope.ownedProjects = function (project) {
        return project.AccessLevel == 1;
    };

    $scope.invitedProjects = function (project) {
        return project.AccessLevel != 1;
    };

    $scope.select = function (project) {
        if ($scope.selected == project)
            $scope.selected = null;
        else
            $scope.selected = project;
    };

    $scope.cancel = function () {
        $modalInstance.close();
    };

    $scope.findProjects = function (projName) {
        $http.get('/api/Project/GetPublicProjectsByName?name=' + projName)
        .success(function (res) {
            $scope.searchResults = res;
        });
    }
});

RDE.controller("SaveItemController", function ($scope, $modalInstance, $cookies) {
    $scope.save = function () {
```

```
        $modalInstance.close();
    };

    $scope.cancel = function () {
        $modalInstance.dismiss('cancel');
    };
});

RDE.controller("SelectThemeController", function ($scope, $modalInstance, $cookies) {
    $scope.theme = $cookies.theme;
    if ($scope.theme == undefined)
        $scope.theme = "cerulean";

    $scope.select = function (val) {
        $cookies.theme = val;
        window.location.reload();
        $modalInstance.close();
    };

    $scope.cancel = function () {
        $modalInstance.close();
    };
});

RDE.controller("AccountController", function ($scope, $http, $window, $state,
AuthorizationService) {
    $scope.registerModel = new AuthorizationService.registerModel();
    $scope.loginModel = new AuthorizationService.loginModel();

    $scope.severError = false;
    $scope.serverMessage = "";

    $scope.GoBack = function () {
        $window.history.back();
    };

    $scope.Register = function () {
        $http.post(
            '/api/Account/Register',
            $scope.registerModel
        ).success(function () {
            AuthorizationService.userLoggedIn();
            $state.go("default");
            $scope.severError = false;
        }).error(function (data) {
            $scope.severError = true;
            $scope.serverMessage = data.Message;
        });
    };

    $scope.Login = function () {
        $http.post(
            '/api/Account/Login',
            $scope.loginModel
        ).success(function () {
            AuthorizationService.userLoggedIn();
            $state.go("default");
            $scope.severError = false;
        });
    };
});
```

```
    }).error(function (data) {
        $scope.serverError = true;
        $scope.serverMessage = data.Message;
    });
};

});

RDE.controller("BaseController",
    function ($scope, $modal, $state, ErrorHandlingService, ProjectService, Item, Tools,
    ItemType, AuthorizationService, modalServices) {
    //Добавя към главния scope user profile-а
    //и после всички останали имат достъп до него
    $scope.userProfile = AuthorizationService.getProfile();
    $scope.project = ProjectService.project;

    $scope.goToItemState = function (e) {
        switch (e.Type) {
            case 2: {
                $state.go("Project.Document", {
                    projectId: $scope.project.ID,
                    ID: e.ID
                });
                break;
            }
            case 3: {
                $state.go("Project.Diagram", {
                    projectId: $scope.project.ID,
                    ID: e.ID
                });
                break;
            }
            default: break;
        }
    };

    $scope.getItem = function (id) {
        var get = function (itemId, collection) {
            if (itemId == -1)
                return $scope.project;
            for (var i = 0; i < collection.length ; i++) {
                if (collection[i].ID == itemId)
                    return collection[i];
                if (collection[i].Type == ItemType["Folder"]) {
                    var search = get(itemId, collection[i].Items);
                    if (search != null)
                        return search;
                }
            }
            return null;
        };

        return get(id, $scope.project.Items);
    }

    $scope.tool = Tools.getTool();
```

```
$scope.addItem = function (e, type) {
    var settings = angular.copy(modalServices.newItemModal);
    settings.resolve = { type: function () { return type; } };
    $modal.open(settings).result.then(function (model) {
        var itemParams = {
            parentId: (e == $scope.project)? -1 : e.ID,
            name: model.Name,
            type: type,
        };
        var templateId = -1;
        if (type == ItemType["Document"] && model.UseTemplate)
            templateId = 1;
        var item = Item.createItem(itemParams, $scope.project.ID, templateId)
            .error(function (data, statusCode) {
                ErrorHandlingService.handleError(data, statusCode);
            })
            .success(function (data) {
                //TODO: тази логика да се измести във service-а който създава item-а
                if (data.Type == ItemType["Folder"]) {
                    data.Items = [];
                    data.opened = true;
                }
                //TODO: тази също
                if (data.Type == ItemType["Diagram"])
                    data.Contents = JSON.parse(data.Contents);
                //Поддържа item-ите
                for (var i = 0 ; i < e.Items.length ; i++) {
                    if (e.Items[i].Type == data.Type) {
                        e.Items.splice(i, 0, data);
                        return;
                    }
                }
                e.Items.push(data);
            });
    });
};

RDE.controller("diagramController", function ($scope, $document, Item) {
    $scope.classDiagrams = function (widget) {
        return widget.type === "class";
    }

    $scope.texts = function (widget) {
        return widget.type === "text";
    };

    $scope.comments = function (widget) {
        return widget.type === "comment";
    }

    $scope.ellipses = function (widget) {
        return widget.type == "ellipse";
    }

    $scope.rectangles = function (widget) { return widget.type == "rectangle";}
```

```
$scope.lines = function (widget) {
    return widget.type === "line";
}

$scope.$on("widget:deleted", function (event, args) {
    var w = args.widget;
    var widgets = $scope.openedDiagram.dirtyContent.widgets;
    for (var i = 0 ; i < widgets.length; i++)
        if (widgets[i] == w) {
            widgets.splice(i, 1);
            break;
        }
});

$scope.$on("create:classDiagram", function (event, args) {
    var classDiagram = Item.createClassDiagram(args.x, args.y);
    $scope.openedDiagram.dirtyContent.widgets.push(classDiagram);
});

$scope.$on("create:comment", function (event, args) {
    var comment = Item.createComment(args.x, args.y);
    $scope.openedDiagram.dirtyContent.widgets.push(comment);
});

$scope.$on("create:text", function (event, args) {
    var text = Item.createText(args.x, args.y);
    $scope.openedDiagram.dirtyContent.widgets.push(text);
});

$scope.$on("create:ellipse", function (event, args) {
    var ellipse = Item.createEllipse(args.x, args.y);
    $scope.openedDiagram.dirtyContent.widgets.push(ellipse);
    $scope.$apply();
});

$scope.$on("create:rectangle", function (event, args) {
    var rectangle = Item.createRectangle(args.x, args.y);
    $scope.openedDiagram.dirtyContent.widgets.push(rectangle);
});

(function () {
    var currentLine = null;
    $scope.$on("tool:changed", function () {
        currentLine = null;
    });
    $scope.$on("line:addPoint", function (event, args) {
        if (!currentLine)
        {
            currentLine = Item.createLine(args.point.x, args.point.y);
            $scope.openedDiagram.dirtyContent.widgets.push(currentLine);
        } else {
            currentLine.addPoint(args.point);
        }
        $scope.$apply();
    });

    $document.on('keypress', function (event) {
        //При натиснат enter
```



```
        if (event.keyCode == 13) {
            currentLine = null;
        }
    });

   })();

});

RDE.controller("MenuController", function ($scope, $rootScope, $modal, $http, modalServices,
AuthorizationService) {
    $scope.newProjectModal = function () {
        if ($scope.userProfile.Logged)
            $modal.open(modalServices.newProject);
    };

    $scope.openProjectModal = function () {
        $modal.open(modalServices.openProject);
    };

    $scope.manageProjectsModal = function () {
        if ($scope.userProfile.Logged)
            $modal.open(modalServices.manageProjects);
    };

    $scope.Logout = function () {
        $http.post('/api/Account/Logout')
            .success(function () {
                AuthorizationService.userLoggedOut();
            })
    };

    $scope.selectThemeModal = function () {
        $modal.open(modalServices.selectTheme);
    };

    $scope.saveItem = function () {
        $rootScope.$broadcast("Item:save");
    }
});

RDE.controller("ProjectExplorerController",
function ($scope, $modal, $http, $state, $rootScope, ProjectService, Item, ItemType,
ErrorHandlingService, modalServices) {
    $scope.openFolder = function (e) { e.opened = true; };
    $scope.closeFolder = function (e) { e.opened = false; }
    $scope.toggleFolder = function (e) { e.opened = !e.opened; }
    $scope.state = $state.current;
    $scope.$on("$stateChangeSuccess", function (event, toState) {
        $scope.state = toState;
    });

    $scope.openItem = function (e) {
        $scope.goToItemState(e);
    }

    //TODO : При отворен документ и изтриване на директория която го съдържа,и ли на същия
документ - да се затваря и документа

```

```
//TODO : Да има Confirm Window.
$scope.delete = function (e) {
    if (e.Type == ItemType["Folder"])
        for(var i = 0 ; i < e.Items.length; i++)
            $scope.delete(e.Items[i]);

    (function () {
        var parent = $scope.getItem(e.ParentID);
        $http.delete('/api/Item/DeleteItem', {
            params: {
                itemId : e.ID
            }
        }).success(function () {
            for (var i = 0 ; i < parent.Items.length ; i++) {
                if (parent.Items[i].ID == e.ID) {
                    parent.Items.splice(i, 1);
                    break;
                }
            }
        }).error(function(data, status, headers, config) {
            ErrorHandlerService.handleError(data,status);
        });
    })();
}

//TODO : Да се добави възможност за cut/copy/paste
//TODO : да се промени директиваната на contextMenu-то за да могат да се добавят глифове
към елемента
$scope.folderOptions = [
    ['Open', $scope.openFolder],
    ['Close', $scope.closeFolder],
    null,
    ['Add Folder', $scope.addItem, ItemType["Folder"]],
    ['Add Document', $scope.addItem, ItemType["Document"]],
    ['Add Diagram', $scope.addItem, ItemType["Diagram"]],
    null,
    ['Delete', $scope.delete]
];

//TODO : Да се добави опция за "View Project info" в която да излизат
//project Owner-a, participants, и някакъв Description
//Премахване на Delete опцията
$scope.projectOptions = $scope.folderOptions.slice(0, $scope.folderOptions.length - 2);

$scope.documentOptions = [
    ['Open', $scope.openItem],
    ['Delete', $scope.delete]
];
$scope.diagramOptions = $scope.documentOptions;
});

RDE.controller("ToolboxController",
function ($scope, Tools) {
    $scope.setTool = function (t) {
        Tools.setTool(t);
    }
    $scope.tool = Tools.getTool();
});
```

```
RDE.controller("WorkingAreaController", function ($scope, $state, $rootScope, $modal, $http,
modalServices, Item, ItemType, ItemInProgress) {

    $scope.ItemsInProgress = [];
    $scope.openedItem = null;
    $scope.openedDocument = null;
    $scope.openedDiagram = null;

    $scope.$on('$stateChangeSuccess', function (event, toState, toParams, fromState, fromParams)
    {

        if (toState.name == "Project.Diagram" || toState.name == "Project.Document") {
            var itemInProgress, id = toParams.ID;
            var item;
            try {
                item = $scope.getItem(id);
            } catch (err) {
                //Колекцията все още не се е заредила
                return;
            }
            if (!isAlreadyProcessed(item)) {
                itemInProgress = new ItemInProgress(item, false);
                $scope.ItemsInProgress.push(itemInProgress);

            } else {
                itemInProgress = findItem(item.ID);
            }
            $scope.openItem(itemInProgress);
        }
    });

    $scope.$on('Item:save', function () {
        $scope.save($scope.openedItem);
    });

    $scope.save = function (item) {
        $http({
            method: "POST",
            url: "api/Item/UpdateItem",
            data: {
                ItemId : item.ref.ID,
                Content: (typeof item.dirtyContent === "string") ? item.dirtyContent :
JSON.stringify(item.dirtyContent)
            }
        })
        .success(function () {
            item.ref.Contents = item.dirtyContent;
        });
    };

    $scope.openItem = function (item) {
        $(document.getElementById("vectorContainer")).children().remove();
        $scope.openedItem = item;
        if (item.ref.Type == ItemType['Document'])
            $scope.openedDocument = item;
        if (item.ref.Type == ItemType['Diagram'])
            $scope.openedDiagram = item;
    };
};
```

```
    }

    $scope.closeItem = function (item) {
        var items = $scope.ItemsInProgress;
        var splice = function (i) {
            items.splice(i, 1);
            if (items[0]) {
                $scope.goToItemState(items[0].ref);
            } else {
                $state.go($state.$current.parent);
                $scope.openedItem = null;
                $scope.openedDocument = null;
                $scope.openedDiagram = null;
            }
        };

        for (var i = 0 ; i < items.length; i++)
            if (items[i] == item) {
                if (item.isDirty())
                    $modal.open(modalServices.saveItemModal).result.then(function () {
                        $scope.save(item);
                        splice(i);
                    }, function () {
                        splice(i);
                    });
                else
                    splice(i);
                break;
            }
    }

    var findItem = function (id) {
        var items = $scope.ItemsInProgress;
        for(var i = 0 ; i < items.length ; i++) {
            if(items[i].ref.ID == id)
                return items[i];
        }
    }

    var isAlreadyProcessed = function (item) {
        var items = $scope.ItemsInProgress;
        for (var i = 0 ; i < items.length; i++)
        {
            if (items[i].ref.ID == item.ID)
                return true;
        }
        return false;
    }

    });

    RDE.directive("classDiagram", function () {
        return {
            restrict: "E",
            scope: {
                diagram : "="
            },
            controller: function ($scope) {
```

```

    $scope.deleteMember = function (e) {
      if (e.type == "prop")
        $scope.diagram.properties.splice(e.index, 1);
      else
        $scope.diagram.methods.splice(e.index, 1);
    }
    $scope.addProperty = function () {
      $scope.diagram.properties.push("");
    };
    $scope.addMethod = function () {
      $scope.diagram.methods.push("");
    };
    $scope.deleteDiagram = function () {
      $scope.$emit("widget:deleted", { widget: $scope.diagram });
    }

    $scope.memberOptions = [
      ['Delete', $scope.deleteMember]
    ];
    $scope.diagramOptions = [
      ['Add property', $scope.addProperty],
      ['Add method', $scope.addMethod],
      null,
      ['Delete', $scope.deleteDiagram]
    ];

  },
  templateUrl : "/Scripts/directives/ClassDiagram/ClassDiagramTemplate.html"
});
});

<svg-wrapper position="diagram.position">
  <div class="ClassDiagram">
    <div class="border-bottom">
      <div
        class="editableField ClassDiagramHeading"
        contenteditable="true"
        ng-model="diagram.title"
        ng-context-menu="diagramOptions">
        <b>{{diagram.title}}</b>
      </div>
    </div>
    <div class="border-bottom">
      <div
        class="editableField"
        contenteditable="true"
        ng-repeat="property in diagram.properties"
        ng-model="diagram.properties[$index]"
        ng-context-menu="memberOptions",
        ng-context-elem="{type:'prop', index:$index}">
        {{property}}
      </div>
    </div>
    <div>
      <div
        class="editableField"
        contenteditable="true"

```

```

        ng-repeat="method in diagram.methods"
        ng-model="diagram.methods[$index]"
        ng-context-menu="memberOptions",
        ng-context-elem="{type:'method', index:$index}">
        {{method}}
    </div>
</div>
</div>
</svg-wrapper>

RDE.directive("comment", function () {
    return {
        restrict: "E",
        scope: {
            comment: "="
        },
        controller: function ($scope) {
            $scope.deleteComment = function () {
                $scope.$emit("widget:deleted", { widget: $scope.comment });
            }

            $scope.commentOptions = [
                ['Delete', $scope.deleteComment]
            ];
        },
        templateUrl: "/Scripts/directives/Comment/CommentTemplate.html"
    }
});

<svg-wrapper position="comment.position">
    <div
        class="Comment"
        ng-context-menu="commentOptions">
        <textarea
            ng-model="comment.text"></textarea>
        <div class="commentFold"></div>
    </div>
</svg-wrapper>

RDE.directive("rdeEllipse", function ($document, $timeout) {
    return {
        restrict: "E",
        templateUrl: "/Scripts/directives/Ellipse/EllipseTemplate.html",
        scope: {
            model : "="
        },
        controller: function ($scope) {
        },
        link: function (scope, element, attrs) {

        }
    };
});

<vector-drag

```

```
resize-offset="{x:35,y:18}"
position="model.position">
<svg>
  <g>
    <ellipse
      ng-attr-cx="{{model.position.x}}"
      ng-attr-cy="{{model.position.y}}"
      ng-attr-rx="{{model.rx}}"
      ng-attr-ry="{{model.ry}}"
      fill-opacity="0.0"
      style="fill: white; stroke: black; stroke-width: 1" />
    </g>
  </svg>
</vector-drag>

RDE.directive("line", function ($document, $timeout, $location, $state) {
  return {
    restrict: "E",
    templateUrl: "/Scripts/directives/Line/LineTemplate.html",
    scope: {
      line: "=",
      points: "="
    },
    controller: function ($scope) {
      var point;
      var startX, startY, x, y, dx, dy;

      $scope.strokeWidth = "2px";
      $scope.thickLine = function ()
      {
        $scope.strokeWidth = "3px";
      }
      $scope.thinLine = function()
      {
        $scope.strokeWidth = "2px";
      }
      $scope.dashArray = "5,5";
      $scope.deleteLine = function () {
        $scope.$emit("widget:deleted", { widget: $scope.line });
        $scope.element.remove();
      }
      $scope.setDashed = function () { $scope.dashArray = "5,5" }
      $scope.setSolid = function() { $scope.dashArray = "0,0" }

      $scope.insertPoint = function (s, elem, event) {
        var pt = {
          x: event.offsetX,
          y: event.offsetY
        }
        var pts = $scope.points;
        for (var i = 0 ; i < pts.length; i++)
        {
          var maxX, maxY, minX, minY;
          maxX = Math.max(pts[i].x, pts[i+1].x);
          minX = Math.min(pts[i].x, pts[i+1].x);
          maxY = Math.max(pts[i].y, pts[i+1].y);
        }
      }
    }
  }
});
```

```
        minY = Math.min(pts[i].y, pts[i+1].y);
        if (pt.x > minX
            && pt.x < maxX
            && pt.y > minY
            && pt.y < maxY) {
            pts.splice(i+1, 0, pt);
            break;
        }
    }
}

$scope.setCircle = function (endPoint) {
    if (endPoint == 'start')
        $scope.line.startPoint = '#markerCircle';
    else
        $scope.line.endPoint = '#markerCircle';
    $scope.$apply();
}

$scope.setArrow = function (endPoint) {
    if (endPoint == 'start')
        $scope.line.startPoint = '#markerBack';
    else
        $scope.line.endPoint = '#markerArrow';
    $scope.$apply();
}

$scope.clearEndPoint = function (endPoint) {
    if (endPoint == 'start')
        $scope.line.startPoint = '';
    else
        $scope.line.endPoint = '';
    $scope.$apply();
}

$scope.endPointOptions = [
    ['Circle', $scope.setCircle],
    ['Arrow', $scope.setArrow],
    ['None', $scope.clearEndPoint]
];

$scope.lineOptions = [
    ['Make dashed', $scope.setDashed],
    ['Make solid', $scope.setSolid],
    ['Insert point', $scope.insertPoint],
    null,
    ['Delete', $scope.deleteLine]
];

$scope.deletePoint = function (i) {
    $scope.points.splice(i+1, 1);
};

$scope.pointOptions = [
    ['Delete', $scope.deletePoint]
];
```



```
$scope.notFirstOrLast = function (pt) {
    var pts = $scope.points;
    return pts[0] != pt && pts[pts.length - 1] != pt;
}

$scope.startDrag = function ($event, pt) {
    point = pt;
    x = pt.x;
    y = pt.y;
    startX = $event.pageX;
    startY = $event.pageY;
    $document.on('mousemove', mousemove);
    $document.on('mouseup', mouseup);
};

function mousemove(event) {
    dy = event.pageY - startY;
    dx = event.pageX - startX;
    point.x = x + dx;
    point.y = y + dy;
    $scope.$apply(function () {
        $scope.calculatePath();
    });
}

function mouseup() {
    $document.off('mousemove', mousemove);
    $document.off('mouseup', mouseup);
}

$scope.calculatePath = function () {
    var path = "M" + $scope.points[0].x + "," + $scope.points[0].y + " ";
    for (var i = 1; i < $scope.points.length; i++) {
        path += "L" + $scope.points[i].x + "," + $scope.points[i].y + " ";
    }
    $scope.path = path;
};

},
link: function (scope, element, attrs) {
    scope.calculatePath();
    scope.$watch("points", function (newValue, oldValue) {
        $timeout(function () {
            scope.$apply(function () {
                scope.calculatePath();
            });
        });
    }, true);
    var vectorContainer = $(document.getElementById("vectorContainer"));
    scope.element = element.children().children();
    vectorContainer.append(scope.element);
    element.remove();
    element.children().remove();
    console.log(scope.line);
}
});
```

```

<svg <!--style="position:absolute; z-index:0;" height="100%" width="100%"--><g>
  <defs>
    <marker id="markerCircle" markerWidth="8" markerHeight="8" refx="5" refy="5">
      <circle cx="5" cy="5" r="3" style="stroke: none; fill: #000000;" />
    </marker>

    <marker id="markerArrow" markerWidth="13" markerHeight="13" refx="2" refy="6"
      orient="auto">
      <path d="M2,2 L2,11 L10,6 L2,2" style="fill: #000000;" />
    </marker>
    <marker id="markerBack" markerWidth="13" markerHeight="13" refx="2" refy="6"
      orient="auto">
      <path d="M10,2 L10,11 L2,6 L10,2" style="fill: #000000;" />
    </marker>
  </defs>
  <path ng-attr-d="{{path}}"
    ng-mouseover="thickLine()"
    ng-mouseout="thinLine()"
    ng-context-menu="lineOptions"
    stroke-dasharray="{{dashArray}}"
    style="stroke: #6666ff; stroke-width: {{strokeWidth}}; fill: none; marker-start:
url('{{line.startPoint}}'); marker-end: url('{{line.endPoint}}');" />
    <circle
      ng-context-menu="endPointOptions"
      ng-context-elem="'start'"
      ng-mousedown="startDrag($event, points[0])"
      ng-attr-cx="{{points[0].x}}"
      ng-attr-cy="{{points[0].y}}"
      r="9"
      style="opacity: 0" />
    <circle
      ng-repeat="point in points | filter:notFirstOrLast"
      ng-mousedown="startDrag($event, point)"
      ng-attr-cx="{{point.x}}"
      ng-attr-cy="{{point.y}}"
      ng-context-menu="pointOptions"
      ng-context-elem="$index"
      r="4"
      stroke="black"
      stroke-width="2"
      fill="white" />
    <circle
      ng-context-menu="endPointOptions"
      ng-context-elem="'end'"
      ng-mousedown="startDrag($event, points[points.length-1])"
      ng-attr-cx="{{points[points.length-1].x}}"
      ng-attr-cy="{{points[points.length-1].y}}"
      r="9"
      style="opacity: 0" />
  </g></svg>

RDE.directive("rectangle", function ($document, $timeout) {
  return {
    restrict: "E",
    templateUrl: "/Scripts/directives/Rectangle/RectangleTemplate.html",
    scope: {
      model : "="
    },
  },

```

```
controller: function ($scope) {
    //var point;
    //var startX, startY, x, y, dx, dy;

    //$scope.strokeWidth = "2px";
    //$scope.thickLine = function ()
    //{
    //    $scope.strokeWidth = "2px";
    //}
    //$scope.thinLine = function()
    //{
    //    $scope.strokeWidth = "2px";
    //}
    //$scope.dashArray = "5,5";
    //$scope.deleteLine = function () {
    //    $scope.$emit("widget:deleted", { widget: $scope.line });
    //    $scope.element.remove();
    //}
    //$scope.setDashed = function () { $scope.dashArray = "5,5" }
    //$scope.setSolid = function() { $scope.dashArray = "0,0" }

    //$scope.lineOptions = [
    //    ['Make dashed', $scope.setDashed],
    //    ['Make solid', $scope.setSolid],
    //    null,
    //    ['Delete', $scope.deleteLine]
    //];

    //$scope.deletePoint = function (i) {
    //    $scope.points.splice(i+1, 1);
    //};

    //$scope.pointOptions = [
    //    ['Delete', $scope.deletePoint]
    //];

    //$scope.notFirstOrLast = function (pt) {
    //    var pts = $scope.points;
    //    return pts[0] != pt && pts[pts.length - 1] != pt;
    //}

    //$scope.startDrag = function ($event, pt) {
    //    point = pt;
    //    x = pt.x;
    //    y = pt.y;
    //    startX = $event.pageX;
    //    startY = $event.pageY;
    //    $document.on('mousemove', mousemove);
    //    $document.on('mouseup', mouseup);
    //};

    //function mousemove(event) {
    //    dy = event.pageY - startY;
    //    dx = event.pageX - startX;
    //    point.x = x + dx;
```

```

        //    point.y = y + dy;
        //    $scope.$apply(function () {
        //        $scope.calculatePath();
        //    });
        //}

        //function mouseup() {
        //    $document.off('mousemove', mousemove);
        //    $document.off('mouseup', mouseup);
        //}
        //$scope.calculatePath = function () {
        //    var path = "M" + $scope.points[0].x + "," + $scope.points[0].y + " ";
        //    for (var i = 1; i < $scope.points.length ; i++) {
        //        path += "L" + $scope.points[i].x + "," + $scope.points[i].y + " ";
        //    }
        //    $scope.path = path;
        //};
    },
    link: function (scope, element, attrs) {
        var vectorContainer = $(document.getElementById("vectorContainer"));
        scope.element = element.children().children();
        vectorContainer.append(scope.element);
        element.remove();
        element.children().remove();
    }
});

<svg>
  <g>
    <ellipse
      ng-attr-cx="{{model.position.x}}"
      ng-attr-cy="{{model.position.y}}"
      ng-attr-rx="{{model.rx}}"
      ng-attr-ry="{{model.ry}}"
      style="fill: white; stroke: black; stroke-width: 1" />
    </g>
  </svg>

RDE.directive("svgWrapper", function ($document, Tools) {
    return {
        restrict: "E",
        transclude: true,
        replace : true,
        scope: {
            position : "="
        },
        templateUrl: "/Scripts/directives/SvgWrapper/SvgWrapperTemplate.html",
        link: function (scope, element, attrs) {
            var startX = 0, startY = 0, x = scope.position.x, y = scope.position.y;
            var tool = Tools.getTool();

            element.css({
                top : scope.position.y + "px",
                left : scope.position.x + "px"
            });

            element.on('mousedown', function (event) {

```

```

        if (tool.name == "Hand") {
            // Prevent default dragging of selected content
            event.preventDefault();
            startX = event.pageX - x;
            startY = event.pageY - y;
            $document.on('mousemove', mousemove);
            $document.on('mouseup', mouseup);
        }
        if (tool.name == "Arrow" && event.which == 3)
            event.preventDefault();
    });

    function mousemove(event) {
        y = event.pageY - startY;
        x = event.pageX - startX;
        scope.position.x = x;
        scope.position.y = y;
        element.css({
            top: y + 'px',
            left: x + 'px'
        });
    }

    function mouseup() {
        $document.off('mousemove', mousemove);
        $document.off('mouseup', mouseup);
    }
}
});

<div
  class="svgWrapper"
  ng-transclude>
</div>

RDE.directive("text", function () {
    return {
        restrict: "E",
        scope: {
            text: "="
        },
        controller: function ($scope) {
            $scope.deleteComment = function () {
                $scope.$emit("widget:deleted", { widget: $scope.text });
            }

            $scope.commentOptions = [
                ['Delete', $scope.deleteComment]
            ];
        },
        templateUrl: "/Scripts/directives/Text/TextTemplate.html"
    }
});

```

```
<svg-wrapper position="text.position">
  <div
    contenteditable
    class="Text"
    ng-model="text.data"
    ng-context-menu="commentOptions">
      {{text.data}}
    </div>
</svg-wrapper>
```

```
RDE.directive("vectorDrag", function ($document, Tools) {
  return {
    restrict: "E",
    transclude: true,
    replace : true,
    scope: {
      position: "=",
      resizeMode : "="
    },
    controller : function($scope) {
      var startX = 0, startY = 0, x, y;
      var tool = Tools.getTool();

      //$scope.opacity = 0;

      //$scope.showResizer = function () {
      //  $scope.opacity = 1;
      //}

      //$scope.hideResizer = function () {
      //  $scope.opacity = 0.1;
      //}

      $scope.translateX = 0;
      $scope.translateY = 0;

      $scope.sX = 1;
      $scope.sY = 1;

      $scope.startDrag = function (event) {
        if (tool.name == "Hand") {
          // Prevent default dragging of selected content
          event.preventDefault();
          startX = event.pageX - $scope.translateX;
          startY = event.pageY - $scope.translateY;
          $document.on('mousemove', mousemove);
          $document.on('mouseup', mouseup);
        }
        if (tool.name == "Arrow" && event.which == 3)
          event.preventDefault();
      };

      function mousemove(event) {
        y = event.pageY - startY;
        x = event.pageX - startX;
        $scope.translateX = x;
      }
    }
  };
});
```

```

        $scope.translateY = y;
        $scope.$apply();
    }

    function mouseup() {
        $document.off('mousemove', mousemove);
        $document.off('mouseup', mouseup);
    }

    $scope.startResize = function (event) {
    },
    templateUrl: "/Scripts/directives/vector_drag/VectorDragTemplate.html",
    link: function (scope, element, attrs) {
        var children = element.find("div").children().children();
        var dragWrapper = element.find("g")[0];
        $(dragWrapper).append(children);
        $(document.getElementById("vectorContainer")).append(element.find("path"));

        $(document.getElementById("vectorContainer")).append(dragWrapper);
        $(element).css({
            display: "none"
        });
    }
});

<div>
    <svg>
        <g class="dragWrapper"
            ng-attr-transform="matrix({{sX}},0,0,{{sY}},{{translateX}},{{ translateY}})"
            ng-mousedown="startDrag($event)">
        </g>
        <path
            opacity="0.1" d="M0,10 L10,0, L10,10, L0,10"
            ng-mousedown="startResize($event)"
            ng-attr-transform="translate({{translateX + position.x + resizeOffset.x}},{{
translateY + position.y + resizeOffset.y}})"></path>
        </svg>
    <div
        class="svgWrapper"
        ng-transclude>
    </div>
</div>

```

```

RDE.directive('contenteditable', function () {
    return {
        require: 'ngModel',
        link: function (scope, element, attrs, ctrl) {
            // view -> model
            element.bind('blur', function () {
                scope.$apply(function () {
                    ctrl.$setViewValue(element.html());
                });
            });
        }
    };
});

```

```
// model -> view
ctrl.$render = function () {
    element.html(ctrl.$viewValue);
};

// load init value from DOM
ctrl.$render();
}
});
});

RDE.directive('ngContextMenu', function ($parse) {
    var renderContextMenu = function ($scope, event, options, elem) {
        if (!$) { var $ = angular.element; }
        $(event.target).addClass('context');
        var $contextMenu = $('<div>');
        $contextMenu.addClass('dropdown clearfix');
        var $ul = $('<ul>');
        $ul.addClass('dropdown-menu');
        $ul.attr({ 'role': 'menu' });
        angular.forEach(options, function (item, i) {
            var $li = $('<li>');
            if (item === null) {
                $li.addClass('divider');
            } else {
                $a = $('<a>');
                $a.attr({ tabindex: '-1' });
                $a.text(item[0]);
                $li.append($a);
                $li.on('click', function (e) {
                    $scope.$apply(function () {
                        item[1].call($scope, elem, item[2], event);
                    });
                });
            }
            $ul.append($li);
        });
        $ul.css({
            display: 'block',
            position: 'absolute',
            left: event.pageX + 'px',
            top: event.pageY + 'px'
        });
        $contextMenu.append($ul);
        $contextMenu.css({
            width: '100%',
            height: '100%',
            position: 'absolute',
            top: 0,
            left: 0,
            zIndex: 9999
        });
        $(document).find('body').append($contextMenu);
        $contextMenu.on("click", function (e) {
            $(event.target).removeClass('context');
            $contextMenu.remove();
        }).on('contextmenu', function (event) {
```



```
        $(event.target).removeClass('context');
        event.preventDefault();
        $contextMenu.remove();
    });
};

return function ($scope, element, attrs) {
    element.on('contextmenu', function (event) {
        $scope.$apply(function () {
            event.preventDefault();
            var options = $scope.$eval(attrs.ngContextMenu);
            var elem = $scope.$eval(attrs.ngContextElem);
            if (options instanceof Array) {
                renderContextMenu($scope, event, options, elem);
            } else {
                throw "'" + attrs.ngContextMenu + "' not an array";
            }
        });
    });
};
});

RDE.directive("diagramView", function (Tools) {
    return {
        scope : {
            diagrams : "="
        },
        restrict: 'A',
        link: function (scope, element, attrs) {
            var $ = angular.element;
            $(element[0]).on("click", function (e) {
                Tools.getTool().click(e);
            });
        }
    };
});

RDE.directive('draggable', ['$document', function ($document) {

    return {
        scope: {
            pos: "="
        },
        link: function (scope, element, attr) {
            var startX = 0, startY = 0, x = 0, y = 0;

            //element.css({
            //    position: 'relative',
            //    border: '1px solid red',
            //    backgroundColor: 'lightgrey',
            //    cursor: 'pointer'
            //});

            element.on('mousedown', function (event) {
                // Prevent default dragging of selected content
                event.preventDefault();
                startX = event.pageX - x;
                startY = event.pageY - y;
```

```
        $document.on('mousemove', mousemove);
        $document.on('mouseup', mouseup);
    });

    function mousemove(event) {
        y = event.pageY - startY;
        x = event.pageX - startX;
        scope.pos.x = x;
        scope.pos.y = y;
        //element.css({
        //    top: y + 'px',
        //    left: x + 'px'
        //});
    }

    function mouseup() {
        $document.off('mousemove', mousemove);
        $document.off('mouseup', mouseup);
    }
}

}

}]]

RDE.directive('resizer', function ($document) {

    return function ($scope, $element, $attrs) {

        $element.on('mousedown', function (event) {
            event.preventDefault();

            $document.on('mousemove', mousemove);
            $document.on('mouseup', mouseup);
        });

        $element.css("height", $attrs.resizerHeight+"px");

        function mousemove(event) {

            if ($attrs.resizer == 'vertical') {
                // Handle vertical resizer
                var x = event.pageX;

                if ($attrs.resizerMax && x > $attrs.resizerMax) {
                    x = parseInt($attrs.resizerMax);
                }

                $element.css({
                    left: x + 'px'
                });

                $($attrs.resizerLeft).css({
                    width: x + 'px'
                });
                $($attrs.resizerRight).css({
                    left: (x + parseInt($attrs.resizerWidth)) + 'px'
                });
            }
        }
    }
});
```

```
    } else {
      // Handle horizontal resizer
      var y = window.innerHeight - event.pageY;

      $element.css({
        bottom: y + 'px'
      });

      $($attrs.resizerTop).css({
        bottom: (y + parseInt($attrs.resizerHeight)) + 'px'
      });
      $($attrs.resizerBottom).css({
        height: y + 'px'
      });
    }
  }

  function mouseup() {
    $document.unbind('mousemove', mousemove);
    $document.unbind('mouseup', mouseup);
  }
};
});

RDE.directive('stopPropagation', function () {
  return {
    restrict: 'A',
    link: function (scope, element, attr) {
      element.bind('click', function (e) {
        e.stopPropagation();
        e.preventDefault();
      });
    }
  };
});

RDE.directive("svgTextInput", function () {
  return {
    restrict: "E",
    replace : "true",
    scope: {
      data: "=",
      x: "=",
      y: "=",
      width: "=",
      height: "=",
    },
    template : '<foreignObject ng-attr-x ng-attr-y ng-attr-width ng-attr-height>'
      + '<body xmlns="http://www.w3.org/1999/xhtml">'
      + '<input type="text" ng-model="data"/>'
      + '</body>'
      + '</foreignObject>'
    }
  });

  //Филтър който връща по подадени проекти
  //списък с проектите на които user-a е 'owner'
```

```
RDE.filter('UserOwnedProjects', function () {
    return function (projects) {
        var result = [];
        for (var i = 0 ; i < projects.length ; i++) {
            if (projects[i].AccessLevel == 1)
                result.push(projects[i]);
        }
        return result;
    }
});

//Филтър който връща по подаден проект и списък с user-и
//списък с user-ите които не участват в този проект
RDE.filter('UsersNotInProject', function () {
    return function (project, users) {
        var result = [];
        for (var i = 0 ; i < users.length; i++) {
            var user = users[i];
            var isInProject = false;
            for (var j = 0 ; j < project.Participants.length; j++)
            {
                var participantName = project.Participants[j].UserName;
                if (user == participantName)
                {
                    isInProject = true;
                    break;
                }
            }
            if (!isInProject)
                result.push(user);
        }
        return result;
    };
});

RDE.factory('Item', function ($http, ItemType) {
    var createContents = function (type) {
        switch (type)
        {
            case ItemType["Document"]: {
                return null;
                break;
            }
            case ItemType["Diagram"]: {
                return JSON.stringify({
                    widgets: []
                });
                break;
            }
            default: break;
        }
    }

    var Item = function (params) {
        this.ID = -1;
        this.ProjectID = -1;
        this.ParentID = params.parentId;
        this.Name = params.name;
    }
});
```

```
        this.Type = params.type;
        this.Contents = createContents(params.type);
    };

    var ClassDiagram = function (x, y) {
        this.type = "class";
        this.title = "Title";
        this.position = { x: x, y: y };
        this.properties = [""];
        this.methods = [""];
    };

    var Text = function (x,y) {
        this.type = "text";
        this.position = { x: x, y: y };
        this.data = "";
    }

    var Comment = function (x, y) {
        this.type = "comment";
        this.position = { x: x, y: y };
        this.text = "Insert text here!";
    };

    var Line = function (x, y) {
        this.type = "line";
        this.position = { x: x, y: y };
        this.points = [];
        this.points.push({ x: x, y: y });
        this.startPoint = '#markerCircle';
        this.endPoint = '#markerArrow';
        this.addPoint = function (point) {
            if (this.position.x < point.x)
                this.position.x = point.x;
            if (this.position.y < point.y)
                this.position.y = point.y;
            this.points.push(point);
        };
    };

    var Ellipse = function (x,y) {
        this.type = "ellipse";
        this.position = { x: x, y: y };
        this.rx = 40;
        this.ry = 20;
    };

    var Rectangle = function (x, y) {
        this.type = "rectangle";
        this.position = { x: x, y: y };
        this.rx = 10;
    }

    return {
        createItem: function (params, projectId, templateId) {
            var item = new Item(params);
            return $http({
                method: "POST",
```

```
        url: "/api/Item/CreateItem",
        data: item,
        params: {
            projectId: projectId,
            templateId: templateId
        }
    });
},
createRectangle : function(x, y) {
    return new Rectangle(x, y);
},
createEllipse : function(x, y) {
    return new Ellipse(x, y);
},
createText : function(x, y) {
    return new Text(x, y);
},
createClassDiagram: function (x, y) {
    return new ClassDiagram(x, y);
},
createComment: function (x, y) {
    return new Comment(x, y);
},
createLine: function (x, y) {
    return new Line(x, y);
}
});

RDE.factory('ItemType', function () {
    return {
        "Folder": 1,
        "Document": 2,
        "Diagram" : 3
    };
});

RDE.factory('ItemInProcess', function () {
    var ItemInProcess = function (itemReference, dirty) {
        this.ref = itemReference;
        this.dirtyContent = angular.copy(itemReference.Contents);
        this.isDirty = function () {
            if (this.ref.Contents == null && this.dirtyContent == "")
                return false;
            return this.dirtyContent != this.ref.Contents;
        }
    }

    return ItemInProcess;
});

RDE.factory('AuthorizationService', function ($http/*, DeserializationService*/) {
    var user = {
        Logged : false
    };

    $http.get('/api/Account/IsUserLogged')
        .success(function (data) {
```

```
        if (data == "true") {
            user.Logged = true;
            syncUserData();
        }
    });

    var syncUserData = function () {
        $.http.get('/api/Account/GetProfile')
            .success(function (data) {
                for (var i in data) {
                    user[i] = data[i];
                }
            });
    };

    var registerModel = function () {
        this.UserName = "";
        this.Password = "";
        this.ConfirmPassword = "";
    };

    var loginModel = function () {
        this.UserName = "";
        this.Password = "";
        this.RememberMe = true;
    };

    return {
        registerModel: registerModel,
        loginModel: loginModel,
        syncUserData: syncUserData,
        getProfile: function () { return user; },
        userLoggedIn: function () { user.Logged = true; syncUserData(); },
        userLoggedOut: function () {
            user.Logged = false;
            user.Projects = [];
        }
    };
});

RDE.factory('DeserializationService', function () {

    var deserializeCircularReferences = function (data) {
        var refs = {}, ids = {};
        fillRefs(data, refs);
        fillIds(data, ids);
        mapRefsToIds(refs, ids);
    };

    var fillRefs = function (data, refs) {
        if (typeof data != "object")
            return;
        for (var i in data) {
            var ref = data[i].$ref;
            if (ref != undefined) {
                if (refs[ref] == undefined)
                    refs[ref] = [];
                refs[ref].push({ parent: data, child: i });
            }
        }
    };
});
```

```
        }
        fillRefs(data[i], refs);
    }
};

var fillIds = function (data, ids) {
    if (typeof data !== "object")
        return;
    if (data.$id !== undefined)
        ids[data.$id] = data;
    for (var i in data) {
        var id = data[i].$id;
        if (id !== undefined) {
            ids[id] = data[i];
        }
        fillIds(data[i], ids);
    }
};

var mapRefsToIds = function (refs, ids) {
    for (var i in refs) {
        for (var j in refs[i]) {
            var parent = refs[i][j].parent;
            var child = refs[i][j].child;
            parent[child] = ids[i];
        }
    }
};

return {
    deserializeCircularReferences: deserializeCircularReferences
};
});

RDE.factory('ErrorHandlingService', function ($http, $modal, modalServices) {
    var handleError = function (data, statusCode) {
        var settings = modalServices.errorModal;
        settings.resolve = {
            data: function () {
                return data;
            }
        };
    };

    switch (statusCode)
    {
        case 400: {
            $modal.open(settings);
            break;
        }
        case 401: {
            data.Message = "Authentication failed. Please log in the system.";
            $modal.open(settings);
            break;
        }
        default: break;
    }
};
});
```



```
        return {
            handleError: handleError
        };
    });

RDE.factory('modalServices', function () {
    //Настройки на модалните прозорци
    var newProject = {
        templateUrl: '/template/modal/newProject',
        controller: 'NewProjectController',
        backdrop: 'static'
    };

    var openProject = {
        templateUrl: '/template/modal/openProject',
        controller: 'OpenProjectController',
        backdrop: 'static'
    };

    var manageProjects = {
        templateUrl: '/template/modal/manageProjects',
        controller: 'ManageProjectsController',
        backdrop: 'static'
    };

    var selectTheme = {
        templateUrl: '/template/modal/selectTheme',
        controller: 'SelectThemeController',
        backdrop: 'static',
        size : 'sm'
    };

    var newItemModal = {
        templateUrl: '/template/modal/newItem',
        controller: 'NewItemController',
        backdrop: 'static',
        size : 'sm'
    };

    var saveItemModal = {
        templateUrl: '/template/modal/saveItem',
        controller: 'SaveItemController',
        backdrop: 'static',
        size : 'sm'
    };

    var errorModal = {
        templateUrl: '/template/modal/error',
        backdrop: 'static',
        size : 'sm',
        controller: function ($scope, $modalInstance, data) {
            $scope.Message = data.Message;
            $scope.close = function () {
                $modalInstance.close();
            }
        }
    };
});
```

```
return {
    newProject: newProject,
    openProject: openProject,
    manageProjects: manageProjects,
    selectTheme: selectTheme,
    newItemModal: newItemModal,
    saveItemModal: saveItemModal,
    errorModal: errorModal
};
});

RDE.factory('ProjectService', function ($http, $rootScope, $state, ItemType) {

    var project = {
        loaded: false,
        opened : true
    };

    var loadProject = function (id) {
        return $http.get('/api/Project/OpenProject', {
            params: {
                id: id
            }
        }).success(function (data) {
            project.loaded = true;
            for (var i in data)
                project[i] = data[i];
            parseContents();
            createTreeStructure();
            if (!$state.$current.includes.Project || !($state.params && $state.params.projectId
== id))
                $state.go('Project', { projectId: id });
            else
                //refresh-ва State-a
                $state.transitionTo($state.current, $state.params, {
                    reload: true,
                    inherit: false,
                    notify: true
                });
        });
    };

    //Преобразува JSON string-а на contents-а към javascript обекти
    var parseContents = function () {
        for (var i = 0; i < project.Items.length ; i++)
            if (project.Items[i].Type == ItemType["Diagram"])
                project.Items[i].Contents = JSON.parse(project.Items[i].Contents);
    };

    //Създава дървовидна структура от линейната която връща server-а
    var createTreeStructure = function () {
        var folders = {};
        for (var i = 0 ; i < project.Items.length ; i++)
        {
            if (project.Items[i].Type == ItemType["Folder"])
            {
```

```
        var items = [];
        project.Items[i].Items = items;
        project.Items[i].opened = true;
        folders[project.Items[i].ID] = items;
    }
}

for(var j = 0 ; j < project.Items.length ; j++)
{
    var item = project.Items[j];
    if (item.ParentID != -1)
    {
        folders[item.ParentID].push(item);
        project.Items.splice(j, 1);
        j--;
    }
}

});

$rootScope.$on('$stateChangeSuccess', function (event, toState, toParams, fromState,
fromParams) {
    if (toParams.projectId) {
        var id = toParams.projectId;
        if (project.ID != id)
            loadProject(id);
    }
});

return {
    project : project,
    loadProject: loadProject,
    unloadProject : function() { project.loaded = false; }
};
});

RDE.factory('Tools', function ($http, $rootScope, $state, ItemType) {
    var $ = angular.element;
    var scale = 1;

    var tool = {
        name: "Arrow",
        click : function() { }
    };

    zoomInClick = function (event) {
        event.stopPropagation();
        scale += 0.15;
        setScale(document.getElementById('diagramContent'), scale);
    };

    zoomOutClick = function (event) {
        event.stopPropagation();
        scale -= 0.15;
        setScale(document.getElementById('diagramContent'), scale);
    };

    var createClassDiagram = function (event) {
```

```
    $rootScope.$broadcast("create:classDiagram", {
      x: event.offsetX,
      y: event.offsetY
    });
  };

  var createComment = function (event) {
    $rootScope.$broadcast("create:comment", {
      x: event.offsetX,
      y: event.offsetY
    });
  };

  var createLine = function (event) {
    $rootScope.$broadcast("line:addPoint", {
      point : {
        x: event.offsetX,
        y: event.offsetY
      }
    });
  };

  var createText = function (event) {
    $rootScope.$broadcast("create:text", {
      x: event.offsetX,
      y: event.offsetY
    });
  };

  var createEllipse = function (event) {
    $rootScope.$broadcast("create:ellipse", {
      x: event.offsetX,
      y: event.offsetY
    });
  };

  var createRectangle = function (event) {
    $rootScope.$broadcast("create:rectangle", {
      x: event.offsetX,
      y: event.offsetY
    });
  };

  var setScale = function (el, scale)
  {
    $(el).css('-webkit-transform', 'scale(' + scale + ')');
    $(el).css('-moz-transform', 'scale(' + scale + ')');
    $(el).css('-ms-transform', 'scale(' + scale + ')');
    $(el).css('transform', 'scale(' + scale + ')');
  }

  return {
    setTool: function (t) {
      tool.name = t;
      $rootScope.$broadcast("tool:changed");
      switch (t) {
        case "Arrow": {
          tool.click = function () { };
        }
      }
    }
  };
}
```

```
        break;
    }
    case "Hand": {
        tool.click = function () { };
        break;
    }
    case "ZoomIn": {
        tool.click = zoomInClick;
        break;
    }
    case "ZoomOut": {
        tool.click = zoomOutClick;
        break;
    }
    case "ClassDiagram": {
        tool.click = createClassDiagram;
        break;
    }
    case "Comment": {
        tool.click = createComment;
        break;
    }
    case "Text": {
        tool.click = createText;
        break;
    }
    case "Line": {
        tool.click = createLine;
        break;
    }
    case "Ellipse": {
        tool.click = createEllipse;
        break;
    }
    case "Rectangle": {
        tool.click = createRectangle;
        break;
    }
    default: break;
}
},
getTool: function() { return tool; }
});
});

(function () {
    window.$ = angular.element;
    window.RDE = angular.module("RDE", ['ui.bootstrap', 'ngAnimate', 'ui.router', 'ngCookies',
    'ui.tinymce']);
    RDE.value('uiTinymceConfig', {
        plugins: "table print preview",
        menubar : 'edit view format table',
        height:440,

    });
    RDE.config(function ($stateProvider, $urlRouterProvider, $httpProvider) {
        $stateProvider
            .state("default", {
```

```
        url: "/",
        templateUrl: "Template/Get/Editor"
    })
    .state('Login', {
        url: '/Login',
        templateUrl: 'Template/Get/Login'
    })
    .state('Registration', {
        url: '/Registration',
        templateUrl: 'Template/Get/Registration'
    })
    .state('Editor', {
        url: "/Editor",
        templateUrl: 'Template/Get/Editor'
    })
    .state('Project', {
        url: "/Project/:projectId",
        templateUrl : 'Template/Get/Editor'
    })
    .state('Project.Document', {
        url: "/Document/:ID",
        //templateUrl : 'Template/Get/Tinymce'
    })
    .state('Project.Diagram', {
        url: "/Diagram/:ID",
        //templateUrl: 'Template/Get/DiagramEditor'
    });
    $urlRouterProvider.when("", function ($state) {
        $state.go("default");
    });

})();

<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Requirements And Design Environment</title>
    @{{ string themeSource = ViewData["theme"] + ".css"; }}
    <link rel="stylesheet" type="text/css" href="/Styles/bootstrap/@themeSource">
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/tinymce")
    @Scripts.Render("~/bundles/angular")
    @Scripts.Render("~/bundles/app")
    @Scripts.Render("~/bundles/app/controllers")
    @Scripts.Render("~/bundles/app/services")
    @Scripts.Render("~/bundles/app/filters")
    @Scripts.Render("~/bundles/app/directives")
    @Scripts.Render("~/bundles/app/models")
  </head>
  <body ng-app="RDE">
    <div ui-view class="height-fill mainView" ng-controller="BaseController">
    </div>
  </body>
</html>
```

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Application name", "Index", "Home", null, new { @class =
"navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                </ul>
            </div>
        </div>
    </div>

    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>

```

```

@model System.Web.Mvc.HandleErrorInfo
@{
    Layout = null;
}

```

```

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Error</title>
</head>
<body>
    <hgroup>
        <h1 class="text-danger">Error.</h1>
        <h2 class="text-danger">An error occurred while processing your request.</h2>
    </hgroup>

```

```

    </hgroup>
</body>
</html>

<nav class="rde-menu navbar navbar-default navbar-static-top" role="navigation" ng-
controller="MenuController">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <a class="navbar-brand" href="#">@*Brand*@</a>
    </div>
    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li dropdown>
          <a class="dropdown-toggle">Project <b class="caret"></b></a>
          <ul class="dropdown-menu">
            <li><a
              ng-class="{ 'disabled-link' : !userProfile.Logged}"
              ng-click="newProjectModal()"><span class="glyphicon glyphicon-
briefcase"></span> New project</a></li>
            <li><a ng-click="openProjectModal()"><span class="glyphicon glyphicon-
folder-open"></span> Open project</a></li>
            <li class="divider"></li>
            <li><a
              ng-class="{ 'disabled-link' : !userProfile.Logged}"
              ng-click="manageProjectsModal()"><span class="glyphicon glyphicon-
cog"></span> Manage my projects</a></li>
          </ul>
        </li>
        <li dropdown>
          <a
            ng-disabled="!userProfile.Logged || !project.loaded"
            class="dropdown-toggle"
            ng-class="{ 'disabled-link' : !userProfile.Logged || !project.loaded}">File
          <b class="caret"></b></a>
          <ul class="dropdown-menu">
            <li><a ng-click="addItem(project,1)"><span class="glyphicon glyphicon-
folder-close"></span> New Folder</a></li>
            <li><a ng-click="addItem(project,2)"><span class="glyphicon glyphicon-
file"></span> New Document</a></li>
            <li><a ng-click="addItem(project,3)"><span class="glyphicon glyphicon-
picture"></span> New Diagram</a></li>
            <li class="divider"></li>
            <li><a ng-click="saveItem()"><span class="glyphicon glyphicon-floppy-
disk"></span> Save</a></li>
          </ul>
        </li>
        <li dropdown>
          <a class="dropdown-toggle" >Options <b class="caret"></b></a>
          <ul class="dropdown-menu">
            <li><a ng-click="selectThemeModal()" >Themes</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
  @*<li dropdown>
    <a

```



```

        class="dropdown-toggle">About <b class="caret"></b></a>
    <ul class="dropdown-menu">
        <li><a ><span class="glyphicon glyphicon-file"></span> Technologies
used</a></li>
        <li><a ><span class="glyphicon glyphicon-picture"></span> Project</a></li>
        <li><a ><span class="glyphicon glyphicon-picture"></span> Author</a></li>
    </ul>
</li>*@
    @*<li dropdown>
        <a class="dropdown-toggle" data-toggle="dropdown">Edit <b
class="caret"></b></a>
        <ul class="dropdown-menu">
            <li><a >Undo</a></li>
            <li><a >Redo</a></li>
            <li><a >Copy</a></li>
            <li><a >Paste</a></li>
        </ul>
    </li>
    <li><a>File</a></li>*@
</ul>
<ul class="nav navbar-nav navbar-right">
    <li ng-if="!userProfile.Logged"><a ui-sref="Login"><span class="glyphicon
glyphicon-user"></span> Login</a></li>
    <li ng-if="userProfile.Logged"><a><span class="glyphicon glyphicon-user"></span>
{{ userProfile.UserName }}</a></li>
    <li ng-if="userProfile.Logged"><a ng-click="Logout()">Log out</a></li>
    @*<li><a><span class="glyphicon glyphicon-comment"></span> Feedback</a> </li>*@
</ul>
</div>
</div>
</nav>

<div class="col-lg-2 height-fill no-padding" ng-controller="ProjectExplorerController">
    <div id="ProjectExplorer"
        @*ng-class="{ 'height-lg-5':state.name=='Project.Diagram'}" *@
        class="panel no-radius panel-default overflow-auto height-fill">
        <div class="panel-heading no-radius">
            <b>Project explorer</b>
        </div>
        <div class="panel-body" >
            <div
                ng-if="project.loaded"
                class="col-lg-12 no-padding projectExplorer">
                <div
                    class="col-lg-12 no-padding pointer"
                    ng-dblclick="project.opened?closeFolder(project):openFolder(project)"
                    ng-context-menu="projectOptions"
                    ng-context-elem="project"><span class="glyphicon glyphicon-
briefcase"></span> {{ project.Name }}</div>
                <div style="clear:both;"></div>
                <ul ng-show="project.opened">
                    <li ng-repeat="node in project.Items"
                        ng-include="'/Template/Get/Node'"></li>
                </ul>
            </div>
        </div>
    </div>
    @*<div id="content-resizer"

```

```

        resizer="horizontal"
        resizer-height="6"
        resizer-top="#ProjectExplorer"
        resizer-bottom="#ProjectProperties">
    </div>*@
    @*<div id="ProjectProperties"
        ng-show="state.name=='Project.Diagram'"
        class="panel no-radius panel-default height-lg-5">
        <div class="panel-heading no-radius">
            <b>Properties</b>
        </div>
        <div class="panel-body">
            Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras aliquam lacinia augue
            at ultrices. Etiam bibendum massa eget mi pretium, ullamcorper varius metus pulvinar. Nulla quis
            semper turpis, ut condimentum metus. Mauris vestibulum tristique tincidunt.
        </div>
    </div>*@

</div>

<div class="col-lg-12 height-fill no-padding">
    <div class="panel height-fill" ng-controller="WorkingAreaController">
        <ul class="nav nav-tabs item-handlers">
            <li ng-repeat="item in ItemsInProcess"
                ng-class="{active:openedItem == item}">
                <a ng-if="item.ref.Type==2"
                    ui-sref="Project.Document({ID:item.ref.ID, projectId : project.ID})">
                    <span class="glyphicon glyphicon-file"></span>
                    {{item.ref.Name}}
                    <span
                        ng-show="item.isDirty()"><strong>*</strong></span>
                    <span
                        class="glyphicon glyphicon-remove pointer"
                        stop-propagation
                        ng-click="closeItem(item);"></span>
                </a>
                <a ng-if="item.ref.Type==3"
                    ui-sref="Project.Diagram({ID:item.ref.ID, projectId : project.ID})">
                    <span class="glyphicon glyphicon-picture"></span>
                    {{item.ref.Name}}
                    <span
                        class="glyphicon glyphicon-remove pointer"
                        stop-propagation
                        ng-click="closeItem(item);"></span>
                </a>
            </li>
        </ul>
        <div ng-show="openedItem.ref.Type==2">
            <textarea id="tinymce" ui-tinymce ng-model="openedDocument.dirtyContent"></textarea>
        </div>

        <div class="diagramEditor" ng-show="openedItem.ref.Type==3">
            <div class="height-fill" ng-include="'/Template/Get/DiagramEditor'"></div>
        </div>
    </div>
</div>

```

```
</div>
```

```
<div>
  <div class="modal-body">
    <alert type="danger">
      <b>There was an error while processing your request.</b>
      <div class="height-1"></div>
      {{ Message }}
    </alert>
  </div>
  <div style="clear:both;"></div>
  <div class="modal-footer">
    <button class="btn btn-primary" ng-click="close()">Ok</button>
  </div>
</div>

<div>
  <div class="modal-header">
    <div class="col-lg-12">
      <div class="btn-group col-lg-3 no-padding" dropdown>
        <button type="button" class="btn btn-default dropdown-toggle">
          Select project <span class="caret"></span>
        </button>
        <ul class="dropdown-menu" role="menu">
          <li ng-repeat="project in userOwnedProjects" ng-click="setSelected(project,
$index)"><a >{{ project.Name }}</a></li>
        </ul>
      </div>
      <div class="col-lg-9">
        <h4 class="col-lg-9 no-padding">{{selected.Name}} <span ng-
if="dirtyModel.isDirty" class="glyphicon glyphicon-asterisk"></span></h4>
        <div class="col-lg-3 no-padding">
          <button
            class="btn btn-default col-lg-12"
            ng-click="deleteProject()"
            ng-disabled="userOwnedProjects.length==0">
            Delete
          </button>
        </div>
      </div>
    </div>
    <div style="clear: both;"></div>
  </div>
  <div class="modal-body">
    <div ng-if="userOwnedProjects.length > 0" class="col-lg-12">
      <div class="row">
        <div class="col-lg-3">
          <b>Name</b>
        </div>
        <div class="col-lg-9">
          <input type="text" ng-model="dirtyModel.Name" ng-change="setDirty()"
class="form-control" placeholder="Project name" />
        </div>
      </div>
      <div class="height-2"></div>
      <div class="row">
```

```

    <div class="col-lg-3">
        <b>Visibility</b>
    </div>
    <div class="col-lg-9">
        <div class="btn-group col-lg-12 no-padding pull-right">
            <label class="btn col-lg-6 btn-success" ng-change="setDirty()" ng-
model="dirtyModel.Visibility" btn-radio="1">Public</label>
            <label class="btn col-lg-6 btn-success" ng-change="setDirty()" ng-
model="dirtyModel.Visibility" btn-radio="2">Private</label>
        </div>
    </div>
</div>
<div class="height-2"></div>
<div class="well">
    @*<div class="col-lg-3">
        <b>Participants</b>
    </div>*@
    <div class="col-lg-12">
        <table class="table">
            <caption>Participants</caption>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Access level</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="participant in dirtyModel.Participants">
                    <td>{{ participant.UserName }}</td>
                    <td>
                        <span ng-if="participant.AccessLevel==1">OWNER</span>
                        <div ng-if="participant.AccessLevel!=1" class="btn-group
col-lg-12 no-padding pull-right">
                            <label class="btn col-lg-4 btn-success" ng-
change="setDirty()" ng-model="participant.AccessLevel" btn-radio="2">E</label>
                            <label class="btn col-lg-4 btn-success" ng-
change="setDirty()" ng-model="participant.AccessLevel" btn-radio="3">E/C</label>
                            <label class="btn col-lg-4 btn-success" ng-
change="setDirty()" ng-model="participant.AccessLevel" btn-radio="4">E/C/D</label>
                        </div>
                    </td>
                    <td>
                        <button ng-if="participant.AccessLevel!=1" class="btn btn-
default" ng-click="removeUser($index)">Remove</button>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
    <div style="clear:both;"></div>
</div>
<div class="height-2"></div>
<div class="row">
    <div class="col-lg-3">
        <b>Search users</b>
    </div>
    <div class="col-lg-9">

```

```

        <div class="input-group">
            <input type="text" ng-model="searchName" placeholder="Search by user
name" class="form-control">
            <span class="input-group-btn">
                <button class="btn btn-default" ng-click="findUsers(searchName)"
type="button">Search</button>
            </span>
        </div>
    </div>

</div>
<div class="row" ng-if="searchResults.length > 0">
    <div class="col-lg-3">
        <b>Search results</b>
    </div>
    <div class="col-lg-9">
        <table class="table">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Access level</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="result in searchResults"
                    ng-init="resultModel={UserName:result, AccessLevel:2};">
                    <td>{{resultModel.UserName}}
                    <td>
                        <div class="btn-group col-lg-12 no-padding">
                            <label class="btn col-lg-4 btn-success" ng-
model="resultModel.AccessLevel" btn-radio="2">E</label>
                            <label class="btn col-lg-4 btn-success" ng-
model="resultModel.AccessLevel" btn-radio="3">E/C</label>
                            <label class="btn col-lg-4 btn-success" ng-
model="resultModel.AccessLevel" btn-radio="4">E/C/D</label>
                        </div>
                    <td>
                        <button class="btn btn-default" ng-
click="addUser(resultModel)">Add</button>
                    <td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
<div ng-if="userOwnedProjects.length==0" class="col-lg-12">
    <h3>You don't own any projects.</h3>
</div>
<div style="clear: both;"></div>
</div>

<div class="modal-footer">
    <div ng-if="showAlert" class="col-lg-12">

```

```

        <alert type="danger">You made changes. Do you want to save the changes before you
continue ?
        <button class="btn btn-default" ng-click="alertClick(true);">Yes</button>
        <button class="btn btn-default" ng-click="alertClick(false);">No</button>
    </alert>
</div>
<button class="btn btn-default" ng-click="close()">Close</button>
<button
    class="btn btn-primary"
    ng-disabled="userOwnedProjects.length==0 || !dirtyModel.isDirty"
    ng-click="save()">Save</button>
</div>

</div>

<div>
    <div class="modal-body">
        <div class="col-lg-12">
            <input type="text" ng-model="model.Name" class="form-control" placeholder="Item
name" />
        </div>
        <div ng-show="type==2" class="col-lg-12">
            <div class="pull-right">
                Use requirements template
                <input ng-model="model.UseTemplate" type="checkbox" />
            </div>
        </div>
    </div>
</div>
<div style="clear:both;"></div>
<div class="modal-footer">
    <button class="btn btn-default" ng-click="cancel()">Cancel</button>
    <button class="btn btn-primary" ng-click="create()">Create</button>
</div>
</div>

<div>
    <div class="modal-header">Create new project</div>
    <div class="modal-body">
        <div class="col-lg-12">
            <alert ng-if="serverError" type="danger" close="closeAlert()"><b>Error!</b> {{
serverMessage }}</alert>
            @*<div ng-if="serverError" class="col-lg-12 alert alert-danger">
                <b>Error!</b> {{ serverMessage }}
            </div>*@
            <div class="col-lg-12 alert alert-warning" ng-if="model.Name.length <
nameMinLength">
                The project name is too short.{{ nameMinLength }} symbols atleast.
            </div>
            <div class="col-lg-12">
                <input type="text" ng-model="model.Name" class="form-control"
placeholder="Project name" />
                <br />
            </div>
        </div>
    </div>
</div>

```

```

    <div class="col-lg-4 pull-left" style="line-height: 38px;">
      &nbsp;    Visibility
    </div>
    <div class="btn-group col-lg-8 pull-right">
      <label class="btn col-lg-6 btn-success" ng-model="model.Visibility" btn-
radio="1">Public</label>
      <label class="btn col-lg-6 btn-success" ng-model="model.Visibility" btn-
radio="2">Private</label>
    </div>
    <div style="clear:both;"></div>
    <br />
    <div class="alert alert-info">The project visibility determines who will be able to
see your project. Public means that it is visible to anyone on the internet. Private means that
only you, and the people who are part of the project can see it's contents.</div>
    </div>
    <div style="clear:both;"></div>
  </div>
  <div class="modal-footer">
    <button class="btn btn-default" ng-click="cancel()">Cancel</button>
    <button class="btn btn-primary" ng-disabled="model.Name.length < nameMinLength" ng-
click="create()">Create</button>
  </div>
</div>

<div>
  <div class="modal-header">Open existing project</div>
  <div class="modal-body">
    <div class="col-lg-12">
      <div class="panel panel-default" ng-if="(projects|filter:ownedProjects).length > 0">
        <div class="panel-heading">My own projects</div>
        <div class="list-group panel-body">
          <a
            @*class="no-radius"*@
            ng-repeat="project in projects | filter:ownedProjects"
            ng-click="select(project)"
            ng-class="{ 'active no-radius': selected==project}"
            class="list-group-item">{{project.Name}}
          </a>
        </div>
      </div>
    </div>
    <div class="height-1"></div>
    <div class="panel panel-default" ng-if="(projects|filter:invitedProjects).length >
0">
      <div class="panel-heading">Other projects(Where I participate)</div>
      <div class="list-group panel-body">
        <a
          @*class="no-radius"*@
          ng-repeat="project in projects | filter:invitedProjects"
          ng-click="select(project)"
          ng-class="{ 'active no-radius': selected==project}"
          class="list-group-item">{{project.Name}}
        </a>
      </div>
    </div>
    <div class="height-1"></div>
    <div class="panel panel-default">

```

```

    <div class="panel-heading">Search projects</div>
    <div class="row">
      <div class="height-1"></div>
      <div class="col-lg-10 col-lg-offset-1">
        <div class="input-group">
          <input type="text" ng-model="searchName" placeholder="Search by
project name" class="form-control">
          <span class="input-group-btn">
            <button class="btn btn-default" ng-
click="findProjects(searchName)" type="button">Search</button>
          </span>
        </div>
      </div>
    </div>
    <div class="list-group panel-body">
      <a
        @*class="no-radius"*@
        ng-repeat="result in searchResults "
        ng-click="select(result)"
        ng-class="{ 'active no-radius': selected==result}"
        class="list-group-item">{{result.Name}}
      </a>
    </div>
  </div>
</div>
<div style="clear: both;"></div>
</div>
<div class="modal-footer">
  <button class="btn btn-default" ng-click="cancel()">Cancel</button>
  <button class="btn btn-primary" ng-disabled="selected == null" ng-
click="open()">Open</button>
</div>
</div>

<div>
  <div class="modal-body">
    <div class="col-lg-12">
      <alert>The item you want to close has been changed. Do you want to save it before
closing it?</alert>
    </div>
  </div>
  <div style="clear:both;"></div>
  <div class="modal-footer">
    <button class="btn btn-default" ng-click="cancel()">Cancel</button>
    <button class="btn btn-primary" ng-click="save()">Save</button>
  </div>
</div>

<div>
  <div style="clear:both;"></div>
  <div class="modal-body">
    <div class="col-lg-6" style="padding-top:8px;">
      <strong>Select theme</strong>
    </div>
    <div class="col-lg-6 no-padding">
      <select class="form-control" ng-model="theme">

```



```

        <option value="amelia">amelia</option>
        <option value="bootstrap">bootstrap</option>
        <option value="cerulean">cerulean</option>
        <option value="cosmo">cosmo</option>
        <option value="cyborg">cyborg</option>
        <option value="darkly">darkly</option>
        <option value="flatly">flatly</option>
        <option value="journal">journal</option>
        <option value="lumen">lumen</option>
        <option value="readable">readable</option>
        <option value="simplex">simplex</option>
        <option value="slate">slate</option>
        <option value="spacelab">spacelab</option>
        <option value="superhero">superhero</option>
        <option value="united">united</option>
        <option value="yeti">yeti</option>
    </select>
</div>
</div>
<div style="clear:both;"></div>
<div class="modal-footer">
    <button class="btn btn-default" ng-click="cancel()">Cancel</button>
    <button class="btn btn-primary" ng-click="select(theme)">Select</button>
</div>
</div>

<div class="no-padding height-fill col-lg-12">
    @Html.Partial("~/Views/Templates/ToolBox.cshtml")
    <div id="diagramView"
        class="col-lg-11 height-fill"
        diagrams="openedDiagram.dirtyContent.widgets"
        ng-controller="diagramController"
        ng-class="{
            CursorHand:tool.name=='Hand',
            CursorZoomIn:tool.name=='ZoomIn',
            CursorZoomOut:tool.name=='ZoomOut',
            CursorClassDiagram:tool.name=='ClassDiagram',
            CursorComment:tool.name=='Comment'}">
        <div
            id="diagramContent"
            class="height-fill"
            diagram-view>
            <svg id="vectorContainer" height="100%" width="100%">

                </svg>
                <class-diagram ng-repeat="diagram in openedDiagram.dirtyContent.widgets |
filter:classDiagrams" diagram="diagram">
                    </class-diagram>
                    <comment ng-repeat="comment in openedDiagram.dirtyContent.widgets | filter:comments"
comment="comment">
                        </comment>
                        <line points="line.points" line="line" ng-repeat="line in
openedDiagram.dirtyContent.widgets | filter:lines"></line>
                            <text text="text" ng-repeat="text in openedDiagram.dirtyContent.widgets |
filter:texts"></text>

```

```

        <rde-ellipse model="ellipse" ng-repeat="ellipse in
openedDiagram.dirtyContent.widgets | filter:ellipses"></rde-ellipse>
    </div>
</div>
</div>

@{
    //Път до елементите
    var p = "~/Views/Templates/EditorElements/";
}
@Html.Partial(p + "Menu.cshtml")
<div class="rde-container container-fluid">
    <div class="row height-fill">
        @Html.Partial(p + "ProjectExplorer.cshtml")
        <div class="col-lg-10 height-fill no-padding">
            @Html.Partial(p + "WorkingArea.cshtml")
        </div>
    </div>
</div>

<div class="container loginPanel">
    <div class="row">
        <div class="col-lg-4 col-lg-offset-4">
            <div class="panel panel-default">
                <div class="panel-heading"><b class="col-lg-offset-1">Log in</b></div>
                <div class="panel-body">

                    <form class="form-horizontal" ng-controller="AccountController">
                        <fieldset>
                            <div class="form-group">
                                <div ng-if="serverError" class="col-lg-10 col-lg-offset-1 alert
alert-danger">

                                    <b>Error!</b> {{ serverMessage }}
                                </div>
                                <div style="clear:both;"></div>
                                <div class="col-lg-10 col-lg-offset-1">
                                    <input type="text" class="form-control" ng-
model="loginModel.UserName" placeholder="Username" />
                                </div>
                                <div class="col-lg-10 col-lg-offset-1 height-1"></div>
                                <div class="col-lg-10 col-lg-offset-1">
                                    <input type="password" class="form-control" ng-
model="loginModel.Password" placeholder="Password" />
                                </div>
                                <div class="col-lg-10 col-lg-offset-1 rememberMePanel">

                                    <div class="pull-right">
                                        Remember me
                                        <input ng-model="loginModel.RememberMe" type="checkbox"

/>

                                    </div>
                                </div>
                                <div class="col-lg-10 col-lg-offset-1 height-2"></div>
                                <div class="col-lg-10 col-lg-offset-1">

```

```

primary col-lg-12">Log in</button>
    <div class="pull-left">
        <a href="" ng-click="GoBack()">Back</a>
    </div>
    <div class="pull-right">
        <a ui-sref="Registration">Register</a>
    </div>
</div>
</div>
</fieldset>
</form>
</div>
</div>
</div>
</div>
</div>
</div>

<div>
    <span
        class="pointer"
        ng-dblclick="(node.Type==1)?toggleFolder(node):openItem(node)"
        ng-context-
menu="(node.Type==1)?folderOptions:(node.Type==2)?documentOptions:diagramOptions"
        ng-context-elem="node">
        <span
            class="glyphicon"
            ng-class="
                {'glyphicon-folder-open': node.Type==1 && node.opened,
                 'glyphicon-folder-close': node.Type==1 && !node.opened,
                 'glyphicon-file': node.Type==2,
                 'glyphicon-picture': node.Type==3}"></span> {{node.Name}}
        </span>
        <ul ng-show="node.opened">
            <li
                ng-repeat="node in node.Items"
                ng-include="'/Template/Get/Node'"></li>
        </ul>
    </div>

<div class="container loginPanel">
    <div class="row">
        <div class="col-lg-4 col-lg-offset-4">
            <div class="panel panel-default">
                <div class="panel-heading "><b class="col-lg-offset-1">Registration</b></div>
                <div class="panel-body">

                    <form class="form-horizontal" ng-controller="AccountController">
                        <fieldset>

                            <div class="form-group">
                                <div ng-if="serverError" class="col-lg-10 col-lg-offset-1 alert
alert-danger">

                                    <b>Error!</b> {{ serverMessage }}
                                </div>

```

```

        <div style="clear:both;"></div>
        <div class="col-lg-10 col-lg-offset-1">
            <input type="text" ng-model="registerModel.UserName"
class="form-control" placeholder="Username" />
        </div>
        <div class="col-lg-10 col-lg-offset-1 height-1"></div>
        <div class="col-lg-10 col-lg-offset-1">
            <input type="password" ng-model="registerModel.Password"
class="form-control" placeholder="Password" />
        </div>
        <div class="col-lg-10 col-lg-offset-1 height-1"></div>
        <div class="col-lg-10 col-lg-offset-1">
            <input type="password" ng-
model="registerModel.ConfirmPassword" class="form-control" placeholder="Repeat password" />
        </div>
        <div class="col-lg-10 col-lg-offset-1 height-3"></div>
        <br />
        <br />
        <br />
        <br />
        <br />
        <div class="col-lg-10 col-lg-offset-1">
            <button type="button" ng-click="Register()" class="btn btn-
primary col-lg-12">Register</button>
            <div class="pull-left">
                <a href="" ng-click="GoBack()">Back</a>
            </div>
        </div>
    </div>
</fieldset>
</form>
</div>
</div>
</div>
</div>
</div>

<div class="col-lg-1 height-fill no-padding rde-toolbox" ng-controller="ToolboxController">
    <div class="btn-group-vertical">
        <button
            ng-class="{active:tool.name=='Arrow'}"
            ng-click ="setTool('Arrow')
            type="button"
            tooltip="Edit widgets"
            tooltip-placement="right"
            class="btn btn-default"><span class="glyphicon glyphicon-arrow-up"></span></button>
        <button
            ng-class="{active:tool.name=='Hand'}"
            ng-click ="setTool('Hand')
            type="button"
            tooltip="Move widgets"
            tooltip-placement="right"
            class="btn btn-default"><span class="glyphicon glyphicon-hand-up"></span></button>
        <button
            ng-class="{active:tool.name=='ClassDiagram'}"
            ng-click ="setTool('ClassDiagram')

```

```

        type="button"
        tooltip="Create class diagram"
        tooltip-placement="right"
        class="btn btn-default"><span class="glyphicon glyphicon-align-
justify"></span></button>
        <button
            ng-class="{active:tool.name=='Comment'}"
            ng-click ="setTool('Comment')"
```

type="button"

tooltip="Create note"

tooltip-placement="right"

class="btn btn-default"></button>

<button

ng-class="{active:tool.name=='Line'}"

ng-click ="setTool('Line')"

type="button"

tooltip="Create Line. Press enter to complete."

tooltip-placement="right"

class="btn btn-default"><i class="vector-path vector-path-line"></i></button>

<button

ng-class="{active:tool.name=='Text'}"

ng-click ="setTool('Text')"

type="button"

tooltip="Create text."

tooltip-placement="right"

class="btn btn-default"><span class="glyphicon glyphicon-text-
width"></button>

<div dropdown class="btn-group">

<button

type="button"

ng-class="{active:tool.name=='Ellipse' || tool.name=='Square'}"

class="btn btn-default dropdown-toggle"

data-toggle="dropdown">

<i class="vector-path vector-path-polygon"></i><span

class="caret">

</button>

<ul class="dropdown-menu">

<a ng-click ="setTool('Square')"><i class="vector-path vector-path-
square"></i>Rectangle

<a ng-click ="setTool('Ellipse')"><i class="vector-path vector-path-
circle"></i>Ellipse

</div>

<div dropdown class="btn-group">

<button

type="button"

ng-class="{active:tool.name=='ZoomIn' || tool.name=='ZoomOut'}"

class="btn btn-default dropdown-toggle"

data-toggle="dropdown">

</button>

<ul class="dropdown-menu">

<a ng-click ="setTool('ZoomIn')"><span class="glyphicon glyphicon-zoom-
in">Zoom in

<a ng-click ="setTool('ZoomOut')"><span class="glyphicon glyphicon-zoom-
out">Zoom out

</div>

</div>
</div>

Използвана литература

1. Ari Lerner ng book The Complete Book on AngularJS
2. Jeffrey Palermo, Jimmy Bogard ASP.NET MVC 4 in Action
3. C# 3.0 Design Patterns O'Reilly
4. Andrew Troelsen Pro C# and the .Net Framework 4.5 Apress
5. <http://www.asp.net/mvc>
6. <http://www.asp.net/web-api>
7. <http://msdn.microsoft.com/en-us/data/ef.aspx>
8. <http://www.tutorialspoint.com/bootstrap/>
9. http://en.wikipedia.org/wiki/Microsoft_SQL_Server
10. http://en.wikipedia.org/wiki/Web_service
11. <http://en.wikipedia.org>
12. Други