|  |  |  |
| --- | --- | --- |
|  | Универзитет „Св. Кирил и Методиј“ во Скопје  Факултет за Информатички Науки и Компјутерско Инженерство  **Институт за интелигентни системи** |  |

**Дипломска Работа**

Ментор: Изработил:

Вон. проф. д-р Слободан Калајџиски Горан Цветков 405/2010

Съдържание

[Апстракт 3](#_Toc428801696)

[1.Вовед 4](#_Toc428801697)

[2.Имплементациски детали 5](#_Toc428801698)

[2.1.Архитектура 5](#_Toc428801699)

[2.1.1.Презентациски слој(MVC Web Application) 7](#_Toc428801700)

[2.1.2.Бизнис сервисен слој(Business service layer) 12](#_Toc428801701)

[2.1.3.Податочно пристапен слој (Data access layer) 14](#_Toc428801702)

[2.2 Bootstrap 19](#_Toc428801703)

[2.3 CSS 20](#_Toc428801704)

[2.4 JavaScript 21](#_Toc428801705)

[3.Professional Video Notes систем 22](#_Toc428801706)

[3.1.Вовед во апликацијата 22](#_Toc428801707)

[3.2.Почетна страница 22](#_Toc428801708)

[3.3.Форма за регистрација 24](#_Toc428801709)

[3.4.Форма за најава 25](#_Toc428801710)

[3.5.Форма за промена на лозинка 25](#_Toc428801711)

[3.6.Dashboard форма за работа со категории, видеа, анотации и 2д графички модели 26](#_Toc428801712)

[3.6.1.Панела за цртање на 2д модел 30](#_Toc428801713)

[3.7.Search механизам 31](#_Toc428801714)

[Заклучок 34](#_Toc428801715)

[Референци 35](#_Toc428801716)

# Апстракт

Апликацијата која е изработена е систем кој служи за анотирање на видео содржини од било каков, но пред се од едукативен карактер. На сите корисници на интернет кои следат онлајн курсеви ќе им служи како алатка преку која ќе може да запишуваат анотации на одредени сегменти од самото видео, за подоцна да можат да се вратат на тоа видео и да пристапат до сегментите од видеото со клик на забелешката која е запишана во тој сегмент од видеото. Исто така корисниците ќе имаат можност да прават пребарувања низ анотациите и видеата кои ги имааат зачувано во нивниот кориснички профил и да ги споделуваат своите анотации со други корисници со цел корисниците кои следат исти курсеви или пак кои се заинтересирани да следат исти курсеви да можат да си помагаат меѓусебно.

Во воведниот дел ќе биде изложена идејата и потребата за ваков систем, зошто е тој корисен, кои се предностите, какви функционалности овој систем нуди. Потоа ќе бидат образложени технологиите кои се користат за изработката на овој систем и како тој е изработен. Понатаму ќе биде прикажано како тој изгледа, како се користи и ќе бидат прикажани неговите функционалности и на крај секако ќе биде даден заклучокот од целиот сработен систем.

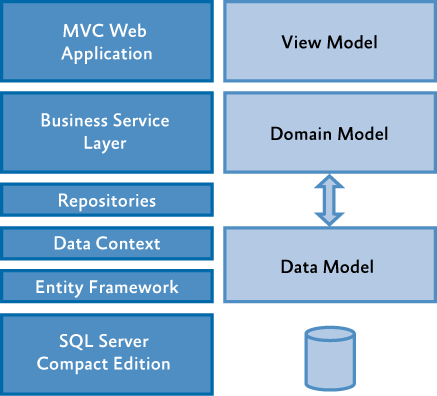
# 1.Вовед

Се повеќе расте бројот на бесплатни курсеви кои ги нудат некои од најпрестижните универзитети во светот како што се Stanford, Yale, MIT, Harvard, Berkeley, Oxford и други. Огромен број на вакви бесплатни курсеви можат да се најдат на youtube каналите на овие универзитети како и на многу други. Софтверското решение приложено во овој дипломски труд има за цел да ја зголеми интерактивноста на студентите кои следат вакви курсеви, а со тоа да го подобрат квалитетот на нивното учење. Professional Video Notes претставува веб апликација на која што корисниците ќе можат да си креираат профил при што ќе имаат можност да создаваат сопствени категории од видеа и во рамките на секое видео ќе внесуваат анотации поврзани за конкретен сегмент од видеото. Ваквото анотирање ќе им овозможи на корисниците лесно и брзо повторување на едукативната содржина приложена во видео материјалите кога ти е ќе имаат потреба од тоа. Откако едно видео ќе биде целосно анотирано, врз база на анотациите корисниците ќе имаат можност да создадат графичка претстава на сето она што го научиле, преку посебна алатка за цртање во 2Д просторот. Ваквата графичка претстава ќе им помогне при добивање на глобална слика за определен едукативен материјал. Корисниците на оваа апликација ќе имаат можност анонимно да ги споделат своите анотации со останатите корисници.

# 2.Имплементациски детали

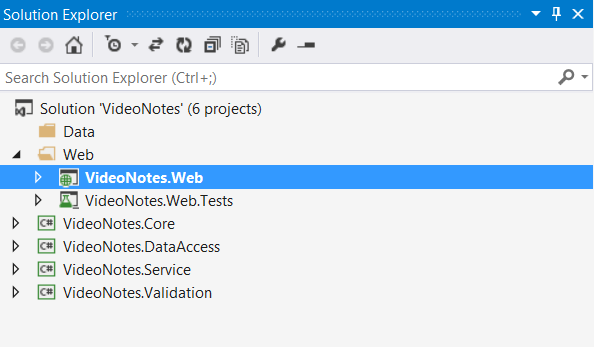
Системот кој е изработен е веб ориентиран и истиот е целосно имплементиран користејќи C# програмен јазик и Microsoft Visual Studio 2013 како развојна околина. Системот е поделен на неколку помали проекти и библиотеки со цел да се задоволат SOLID([**Single responsibility principle**](https://en.wikipedia.org/wiki/Single_responsibility_principle), [**Open/closed principle**](https://en.wikipedia.org/wiki/Open/closed_principle), [**Liskov substitution principle**](https://en.wikipedia.org/wiki/Liskov_substitution_principle), [**Interface segregation principle**](https://en.wikipedia.org/wiki/Interface_segregation_principle), [**Dependency inversion principle**](https://en.wikipedia.org/wiki/Dependency_inversion_principle)) принципите. Презентацискиот слој е изработен во ASP.NET MVC 5 технологија.

## 2.1.Архитектура



Слика 3.1-1. Back-end архитектура на системот

MVC Web Application го претставува презентациското ниво на системот и истото е одговорно да ги обработува HTTP барањата на клиентите, да ги мапира влезните и излезните податоци во View Models и истите да ги прикажува на клиентите преку таканаречените View-а. MVC Web Application исто така има за цел да воведе валидација на клиентска страна, да ги мапира View Models во Domain Models и како такви да ги испрати на Business Service Layer. Business Service слојот ја содржи целата бизнис логика на апликацијата и истиот има за цел да врши валидација на серверска страна. Ваквата структура овозможува лесно тестирање на одделните модули.Repositories подслојот ни дава уште еден степен на апстракција врз работата со податоци од базата. Целта на ова ниво е да ни овозможи апстракција на работата со основните CRUD операции со податоците од базата. Data context-от е извор на сите ентитети мапирани врз конекцијата на базата на податоци, односно ги проследува сите промени кои што се прават врз ентитетите и одржуваат “identity cache” кој што гарантира дека ентитетите коишто се превземаат повеќе од еден пат се репрезентирани со користење на иста инстанција. Entity Framework е open source framework за објектно- релационо мапирање. Еntity framework-от им овозможува на програмерите да работат по апстрактно кога работат со податоци и да може да создаваат податочно- ориентирани апликации со помалку код за разлика од традиционалните апликации. Microsoft SQL Server Compact е компактна релациона база на податоци за апликации кои што се извршуваат на мобилни и десктоп уреди. Microsoft SQL Server споделува заедничко API со другите Microsoft SQL Server изданија. Таа исто така влучува [ADO.NET](http://ado.net/" \t "_blank) провајдери за пристап до податоци користејќи [ADO.NET](http://ado.net/" \t "_blank) API-ја и вградени можности за синхронизација , како и поддршка за LINQ и Entity Framework. SQL CE базите на податоци можат да бидат креирани и менаџирани од Microsoft Visual Studio, како и од некои постари верзии на SQL Server Management Studio.



Слика 3.1-2. Повеќе слојна архитектура на ProVideoNotes системот

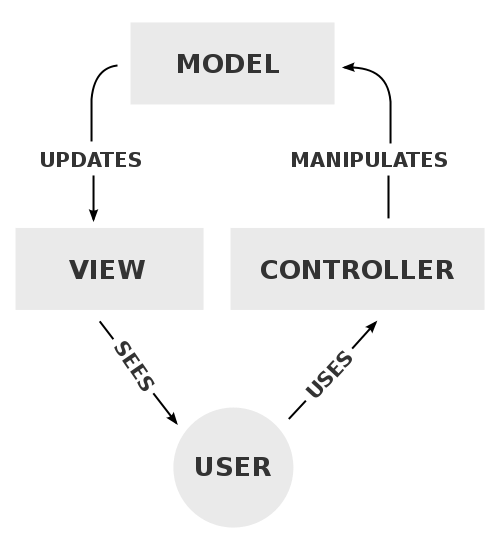
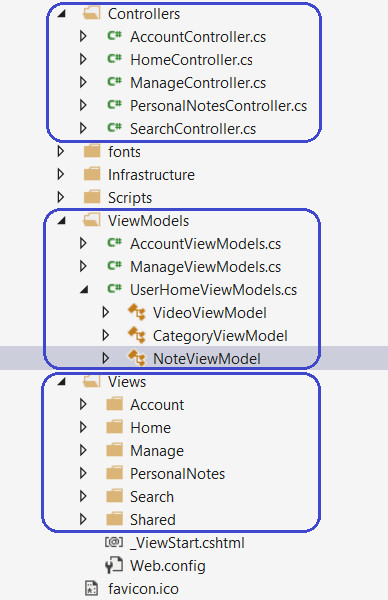
### 2.1.1.Презентациски слој(MVC Web Application)

Microsoft ASP.NET е изграден на .NET платформата, која обезбедува API (Application Program Interface) интерфејс за софтверските програмери да креираат апликации. Тие апликации можат да бидат веб базирани или десктоп апликации. Со цел ASP.NET веб сајтот да функционира правилно мора да биде објавен на веб сервер што подржува ASP.NET апликации. Microsoft's Internet Information Services (IIS) е веб серверот што најчесто се користи и што е искористен при креирање на апликацијата.

MVC(Model-View-Controller) е софтверски дизајн патерн за имплементирање на кориснички интерфејси. Тој ја дели апликацијата на три меѓусебно поврзани дела, со цел да ги раздели внатрешната репрезентација на информацијата од начинот на кој таа информација е прикажана или примена од страна на корисникот.

Врз основа на ASP.NET, ASP.NET MVC им овозможува на програмерите да изградат веб апликација која се состои од три делови: Модел, Приказ и Контролер, односно апликацијата е поделена на три логички слоеви:

* Модел (Бизнис слој)
* Приказ (Презентациски слој)
* Контролер (Слој кој ги поврзува моделот со приказот)



2.1.1-1. Имплементацијата на MVC pattern во VideoNotes системот

#### 2.1.1.1.Model

Моделот ја содржи бизнис логиката, управува со состојбите и нуди методи за пристап и мутирање на состојбата на апликацијата. Модел слојот типично е поделен на неколку подслоја:

Service Layer – овој слој обезбедува конкретна работа, логика на високо-ниво на определени делови од апликацијата. Овој слој е дирекно повикуван од контролерите и помошните “view” класи.

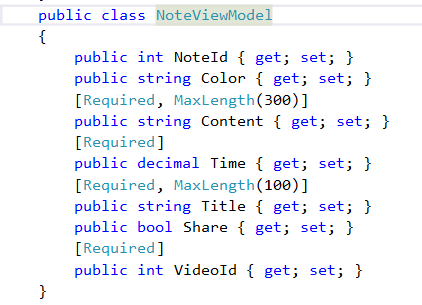
Data Access Layer – овој слој нуди пристап до базата на податоци и се повикува исклучиво од страна на Сервис слојот. Објектите во овој слј не знаат едни за други.

Value Objects Layer – овој слој овозможува едноставна, дата ориентирана репрезентација на крајните јазли во хиерархијата на моделот.

Архитектурата на моделот не е добро направена ако:

* Содржи сесиска логика
* Value објектите референцират кон сервис објектите или gateway објектите

Ако моделот содржи некои од овие работи тоа е знак дека моделот го руши принципот на енкапсулација. Со овозможувањето на сервисниот слој да референцира кон сесиите исто така се креира модел кој што е тежок за тестирање без да ја има целата апликација. Затоа чувањето на сесискте променливи во моделот прави тој да биде многу тешко реискористуван од другите контролери.



Слика 2.1.1.1-1 View Model кој гo опишува моделот на една анотација(note)

Контролерот користејќи го овој модел ги прифаќа податоците од HTML POST барањето пратени од корисникот, кои притоа се валидираат од страна на Data Annotation атрибутите поставени над соодветните својства (пр. [Required, MaxLenght(100)]). Data Annotation атрибутите вршат валидација како на серверска страна така и на клиентска страна со инјектирање на javascript валидација при рендерирањето на приказите(view-a).

#### 2.1.1.2.View

Работата на овој слој е да ги рендерира податоците кои му се прикажуваат на клиентот. Податоците главно се обезбедени од контролерот, но и view-то може да има податоци кои не се поврзани со моменталното барање (aside data, footer data).

Архитектурата на view-то не е добро направена ако:

* Содржи бизнис логика
* Содржи сесиска логика

Ако view-то содржи некои од овие работи тоа е знак дека view слојот прави многу нешта. Бизнис логиката треба да биде вметната во Сервис објектот, а сесиските референци во контролерот. Ова ќе овозможи view-to да биде лесно вградливо и тестирано затоа што тие пред се се засноваат на податоци, а не на целосната логика на апликацијата.

##### 2.1.1.2.1.ASP.NET Razor

Razor e ASP.NET синтакса која се користи за креирање динамички веб страници со C# и Visual Basic програмските јазици. Razor е развиен во Јуни 2010 година и беше пуштен во употреба за Microsoft Visual Studio 2010 во Јануари 2011 година. Razor е многу едноставен engine за view-а и беше пуштен во употреба како дел од ASP.NET MVC 3 и Microsoft WebMatrix сетот на алатки. Синтаксата на Razor e template базиранa на C# јазикот, која им овозможува на програмерите да користат HTML конструкција. Наместо да користат .ASPX  од ASP.NET синтакса со <%= %> симболи за да прикажат блокови од кодови, Razor синтаксата започнува со @ и не бара експлицитно затварање на блокот на кодот.

Предности на Razor

Идејата зад Razor е да овозможи оптимизирана синтакса за HTML генерациите користејќи код-фокусиран template пристап со минимални транзиции помеѓу HTML кодот и кодот. Овој дизајн го намалува бројот на карактери и овозможува полесна работа при кодирање, без барања експлицитно да не се користат серверски блокови во HTML кодот. Исто така предности на Razor се:

* Поддржува  [IntelliSense](https://en.wikipedia.org/wiki/IntelliSense)
* Може да се unit-тестира
* Поддржува layouts



Слика 2.1.1.2-1. Користење на Razor при имплементација на view

#### 2.1.1.3.Controller

Работата на контролерот е да ги преведува влезните побарувања во излезни одговори. За да го направи тоа контролерот мора да ги земе податоците од барањето и да ги пренесе до Сервис слојот. Потоа сервис слојот враќа податоци кој што контролерот ги дава на View-то за тој да ги рендерира.View-то може да биде HTML за стандардно веб барање или нешто како JSON за RESTful API барање.

Архитектурата на контролерот не е добро направена ако тој:

* Прави многу барања до сервис слојот
* Прави многу барања до сервис слојот кои што не враќаат податоци
* Прави барања до сервис слојот без да подава аргументи

Ако контролерот прави некои од овие посочени работи тогаш тоа е знак дека контролер слојот прави многу нешта. За да тоа се поправи, треба да се рефакторира логиката со цел да се направи сервис слој кој што ќе врши конкретна работа и ќе може да биде повикуван од контролерот. Уште повеќе, ако контролерот прави многу барања без да праќа аргументи, тогаш тоа ни укажува дека сервис слојот не ги следи принципите на енкапсулација.

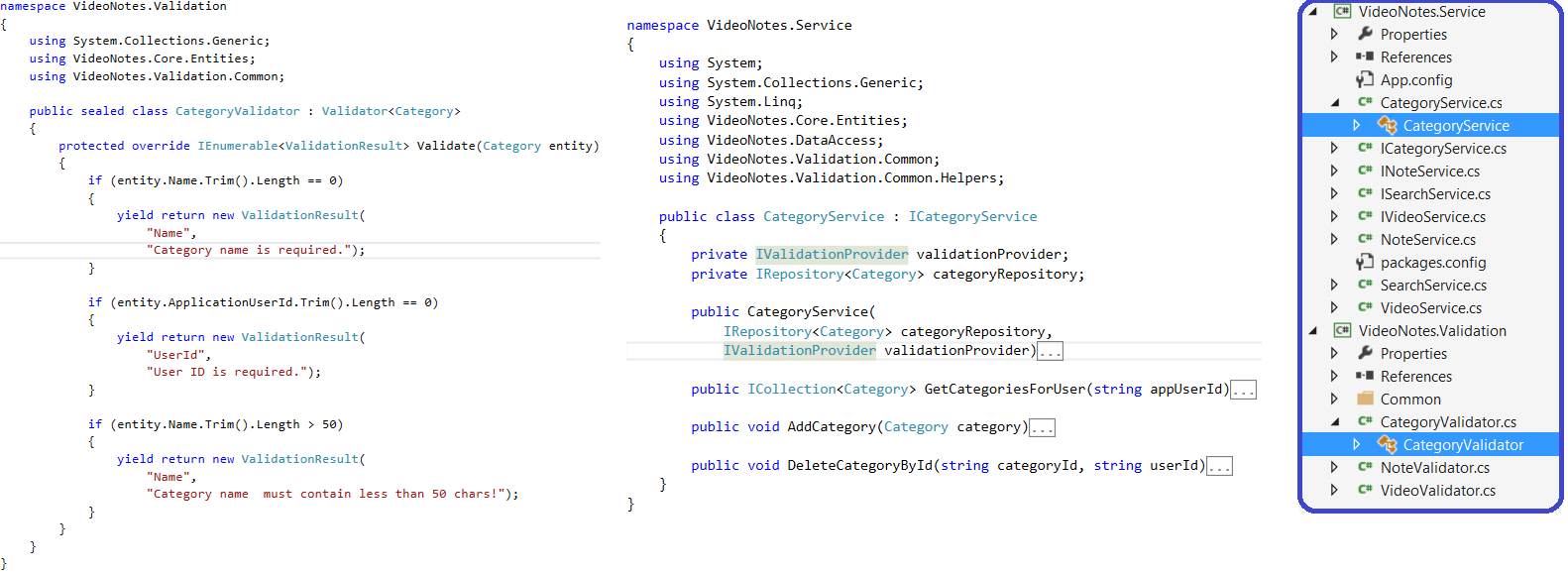


Слика 2.1.1.3-1. AddCategory акцијски метод

### 2.1.2.Бизнис сервисен слој(Business service layer)

Кога во апликацијата има многу бизнис логика тогаш треба да се креира ваков бизнис слој. Овој слој е уште еден слој на апстракција и затоa вреди да се додаде при имплементацијата на апликации. Креирањето на ваков слок овозможува да се тестира овој слој пооделно и независно од другите слоеви и исто така ги упростува тестовите за контролерите. Бидејќи бизнис слојот е на страна од корисничкиот интерфејс, тој исто така може да се користи во иднина доколку сакаме да нудиме веб сервиси. Ова овозможува desktop и mobile клиенти на еден бизнис слој.

Кога правиме одлука дали да креираме бизнис слој, треба да земаме во предвид дали да креираме и посебен domain model. Креирањето на одделни domain модели заедно со бизнис слојот нуди многу бенефити кога треба целосно да се енкапсулира податочниот модел, кога податочниот модел не нуди валидација и кога сакаме да ја олесниме работата при пишување контролерите и view-та. Сепак кога креираме ваков слој треба да имаме предвид дека имаме некаква цена за трансфер меѓѕ двата модели. Сервис слојот примарно содржи handler-и и domain модели. Handler-ите се сет на класи кои што го имплементираат главното јадрено однесување на апликацијата. Тие се сосема независни и на страна од корисничкиот интерфејс. Domain моделите се втор сет на класи во сервисниот слој кои што се разликуваат од податочните модели и од view моделите и не се грижат за корисничкиот интерфејс. Handler-ите заедно до Domain моделите ја претставуваат бизнис логиката на аплиакцијата. Заедно тие овозможуваат валидација, пресметка на статистики и други сервиси. Целокупно бизнис сервис слојот овозможува функционалности кои го прават пишувањето на контролерите, акциите и view-та многу полесно.



Слика 2.1.2-1. Сервис кој ја содржи бизнис логиката за работа со категории

На сликата е претставен сервисот кој овозможува услуги на контролерот за работа со категории. Податоците од контролерот минуваат преку валидациите овозможени од страна на слојот за валидирање и се предаваат на сервисот кој има за цел да овозможи запишување на истите во базата на податоци користејќи го репозитори слојот.

### 2.1.3.Податочно пристапен слој (Data access layer)

Слика 2.1.3-1. Имплентација на податочно пристапен слој

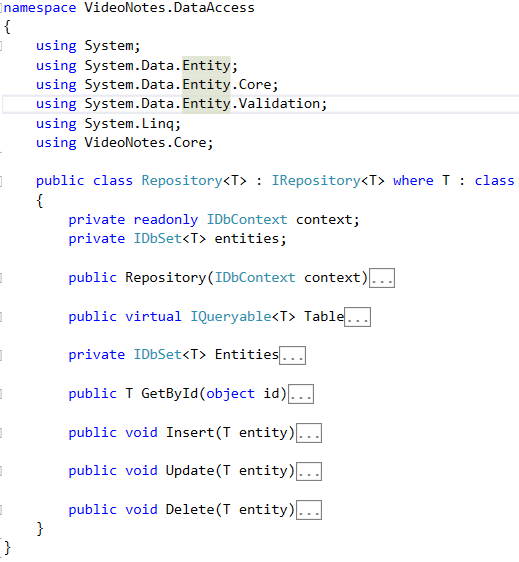
Овој слој има за цел да овозможи упростен пристап до податоците од базата на податоци. DAL ги мапира табелите или определени записи од истите во референци кон објекти кои му овозможуваат на програмерот да креира модули со високо ниво на апстракција. Сервисниот слој ги користи интерфејсите на DAL за да запишува и чита од базата на податоци.

#### 2.1.3.1.Repository pattern

Repository pattern помага да се оддели базата на податоците од бизнис логиката на апликацијата. Овој шаблон е корисен особено кога се користи  ADO.NET Entity Framework затоа што:

* овозможува да се скријат специфични класи од ADO.NET Entity Framework –от како на пример DbContext и DbSet
* овозможува да се оптимизира формата на податоци вратени од апликацијата,
* координира update-ти
* овозможува лесно тестирање на апликацијата без притоа да биде потребна физичка меморија.

Repository pattern-от е сет на интерфејси и имплементации кои што овозможуваат методи за пристап до податоци. Колко repository-а треба да се креираат, се заснова на тоа колко грануларно сакаме да го направиме пристапот до податоци во апликацијата.



Слика 2.1.3.1-1. Имплементација на repository pattern во DAL на ProVideoNotes

#### 2.1.3.2.Data context

Data context-от е извор на сите ентитети мапирани врз конекцијата на базата на податоци, односно ги проследува сите промени кои што се прават врз ентитетите и одржуваат “identity cache” кој што гарантира дека ентитетите кои што се превземаат повеќе од еден пат се репрезентирани со користење на иста инстанција. Data context е лесен за користење и неговото креирање не чини многу скапо. Типична LINQ to SQL апликација креира Data context инстанција во scope на метод или класа која живее кратко која што репрезентира логички сет на операции од базата на податоци.

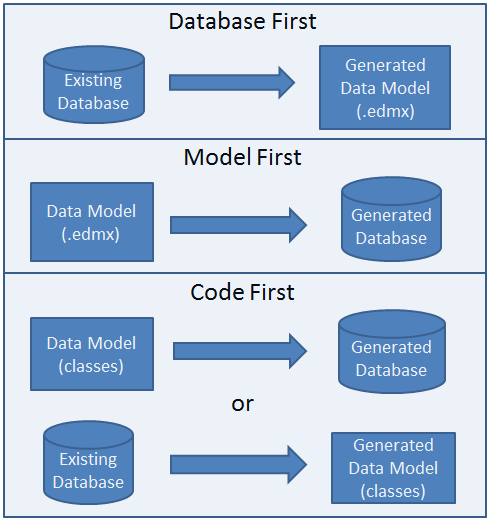
Слика 2.1.3.2-1. Имплементација на Data context кај ProVideoNotes

#### 2.1.3.3.Entity framework

За поврзување и работа со податоците од базата на податоци во апликацијата се користи Entity Framework 5.0 кој што е open source framework за објектно- релационо мапирање. Овој framework е сет од технологии во ADO.NET кој што овозможува развивање на податочно-ориентирани апликации. Архитектите и програмерите на податочно- ориентирани апликации обично наоѓаат тешкотии од потребата да постигнат две многу различни цели. Тие мораат да ги моделираат ентитетите, врските и логиката на бизнис проблемите кои ги решаваат и исто да работат со engine-и за манипулирање на податоци за да зачувуваат и да земаат податоци. Entity Framework-от им овозможува на програмерите да работат со податоците како со специфични објекти, елиминирајќи ја потребата од пишувањето на код при нормална работа со податоците од базата на подтоци. Еntity framework-от им овозможува на програмерите да работат по апстрактно кога работат со податоци и да може да создаваат податочно- ориентирани апликации со помалку код за разлика од традиционалните апликации. Бидејќи Entity Framework-от е дел од .NET, апликациите кои го користат овој framework може да се извршуваат на секој компјутер кој што има инсталирано .NET Framework.

Постојат неколку сценарија во кои Entity Framework е корисен:

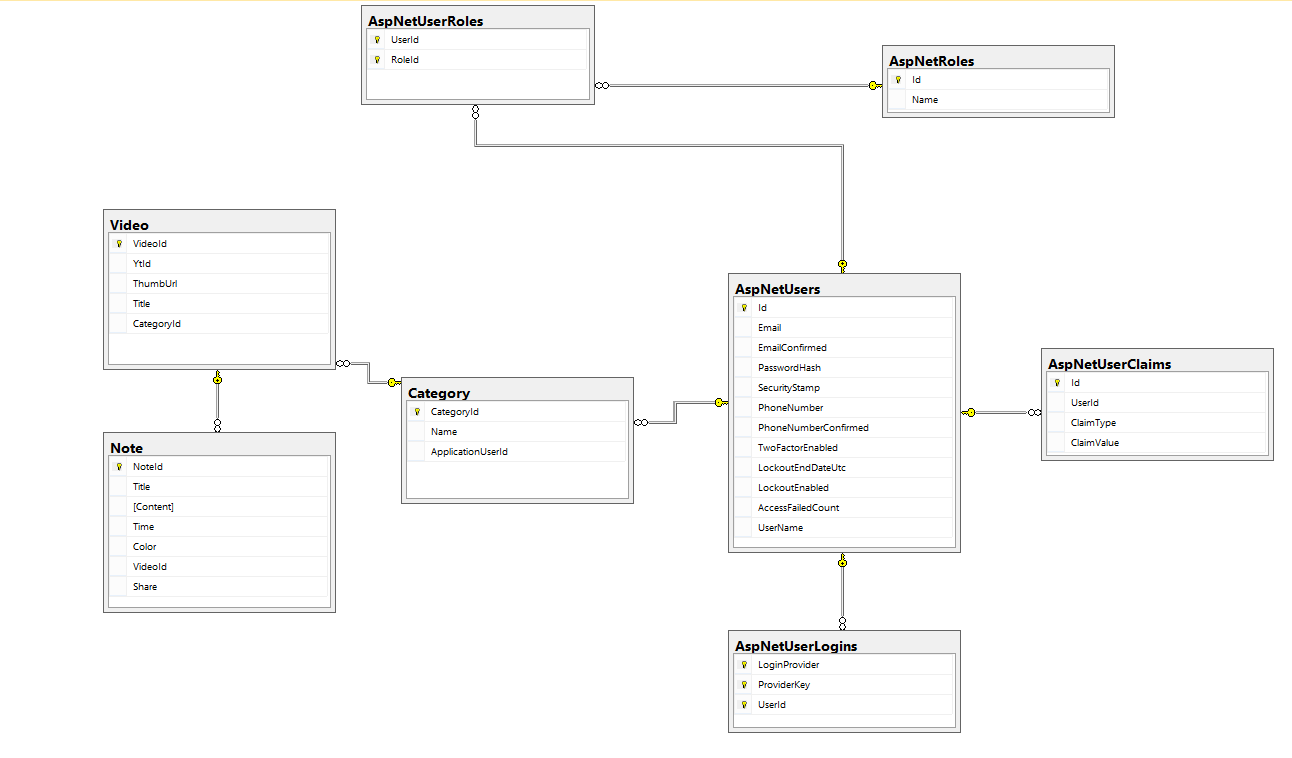
* Прво, ако веќе имате постоечка база на податоци или ако сакате да ја креирате базата пред другите делови од апликацијата.
* Второ ако се креирани класи во апликацијата и самите табели во базата да ви претставуваат тие класи.
* Трето, ако сакате да го дизајнирате вашиот дијаграм на базата на податоци, и потоа да ја креирате самата база и класите.

 Слика 2.1.3.3-1. Сценарија кај Entity Framework

#### 2.1.3.4 SQL Server

SQL SERVER

Microsoft SQL Server Compact е компактна релациона база на податоци за апликации кои што се извршуваат на мобилни и десктоп уреди. Microsoft SQL Server споделува заедничко API со другите Microsoft SQL Server изданија. Таа исто така влучува ADO.NET провајдери за пристап до податоци користејќи ADO.NET API-ја и вградени можности за синхронизација , како и поддршка за LINQ и Entity Framework. SQL CE базите на податоци можат да бидат креирани и менаџирани од Microsoft Visual Studio, како и од некои постари верзии на SQL Server Management Studio.

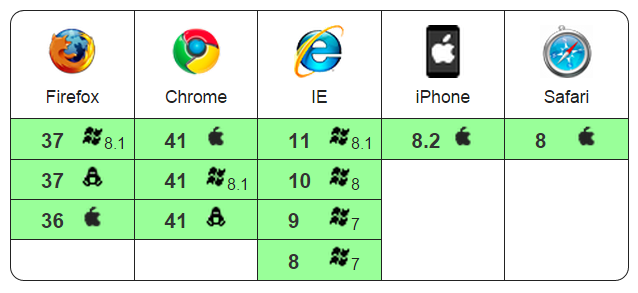


Слика 2.1.3.4-1. Дијаграм на базата на податоци

## 2.2 Bootstrap

Bootstrap е open source framework за креирање на responsive design веб страни и веб апликации со цел да ја олесни работата при веб развивањето.Bootstrap e front end платформа и претставува интерфејс помеѓу корисникот и кодот на страна на серверот и е дизајниран за да го поддржи развојот на веб апликации. Bootstrap е модуларен и се состои од серија на LESS stylesheets кои ги имплементираат различните toolkit компоненти. Програмерите можат да ја адаптираат и самата Bootstrap датотека, со избирање на компонентите кои сакаат да ги користат во нивниот проект.

Изграден е така што да работи максимално најдобро на било каков пребарувач.Тој ги поддржува последните верзии на следните пребарувачи и платформи.



Слика 2.2.1. Bootstrap поддршка за различни верзии на пребарувачи

Благодарение на овој моќен framework изгледот на апликацијата на различни уреди, различно се рендерира во зависнот од резолуцијата на екранот на уредот.

## 2.3 CSS

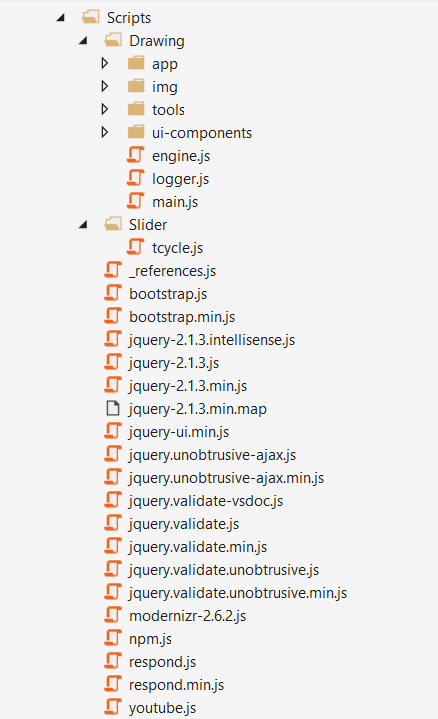
CSS е кратенка за Cascading Style Sheets, кои претставуваат карактеристика додадена на HTML што им дава на развивачите на веб-страници и корисниците поголема контрола врз приказот на веб страниците. CSS примарно е дизајниран за да овозможи поделба на содржината на документот од презентацијата, вклучувајќи елементи како што се бои, фонтови итн. Оваа поделба овозможува поголема флексибилност и контрола во спецификацијата на презентациските карактеристики, со што овозможува повеќе HTML страници да споделуваат форматирање преку специфицирање на релевантен CSS во посебен .css фајл.

Bootstrap обезбедува сет на stylesheets со што обезбедува модерен и уникатен стил за сите HTML елементи односно се состои од серија на LESS stylesheets кои ги имплементираат различните toolkit компоненти. LESS е CSS препроцесор, што значи дека го проширува CSS, додавајќи карактеристики кои им овозможуваат на променливите, функциите многу техники кои овозможуваат CSS да биде по одржлив и со можност за проширување.

## 2.4 JavaScript

JavaScript е скриптирачки јазик кој се користи како дел од веб прелистувачите, чија имплементација овозможува client-side скриптите да комуницираат со корисникот и да го контролираат приказот на содржината на апликацијата. Javascript може да комуницира со HTML, овозможувајќи да се креираат сајтови со динамички содржини. JavaScript е одобрен од страна на голем број на софтверски компании и е отворен јазик кој може да се користи без купување на лиценца. jQuery е бесплатна Javascript библиотека , која се користи за навигација на HTML документи, справување со настани, создавање на анимации и додадавање на Ajax интеракции на веб страниците.

Bootstrap доаѓа со неколку JavaScript компоненти во форма на jQuery plugins. Тие овозможуваат елементи за корисничкиот интерфејс како на пример dialog boxes, tooltips и carousels. Тие исто така ја прошируваат функционалноста на некои од постоечките интерфејс елементи.

 Слика 2.4-1. JavaScript користени скрипти

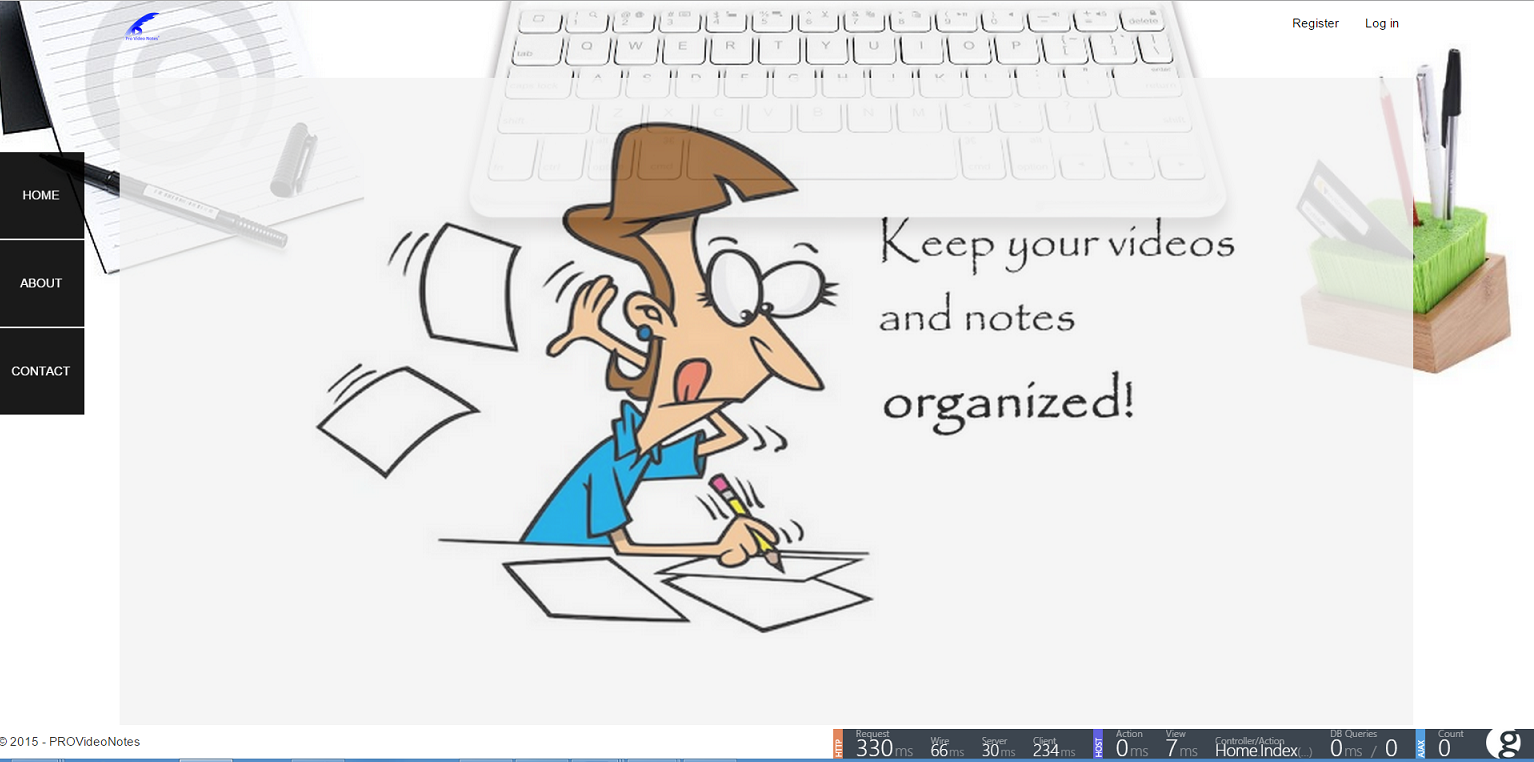
# 3.Professional Video Notes систем

## 3.1.Вовед во апликацијата

Системот е изработен со едукативни цели, односо за луѓето кои следат online курсеви на сајтот на youtube.com. Се повеќе универзитетите, курсевите кои ги водат ги прикажуваат online со цел поголема маса на луѓе да имаат пристап до нив. Водејќи се од овој факт, изработив систем кој ќе помага на корисниците на ваков тип на курсеви, со тоа што ќе им понуди апликација преку која ќе може да ги следат курсевите, да запишуваат белешки на одредени сегменти од видео записот, за подоцна да се вратат на сегментот од видеото за кој што има запишано белешка и истите да ги споделат со другите корисници и да скицираат модели кои ќе бидат запишани во форма на слики на локалната машина. Системот исто така им овозможува овие видео-курсеви или записи да ги поделат во различни категории со цел да имаат подобра организација на курсевите кои ги следат. Пребарувачкиот механизам имплементиран во системот овозможува брзо и лесно пребарување по неколку различни критерии:

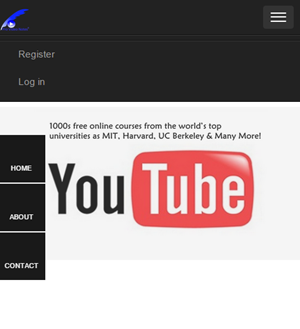
1. Пребарување по лични анотации
2. Пребарување по наслови на личните видео записи
3. Пребарување по споделени анотаци
4. Пребарување по наслови на сите видео записи
5. Пребарување по youtube линк

## 3.2.Почетна страница



Слика 3.1-1. Почетна страна десктоп верзија

На сликата е претставена почетната страница на системот. Истата содржи slide show на кој минуваат неколку intro слики на кои е претставена основната идеа на системот. Од левата страна на сликата се гледа основното мени за навигирање по страната



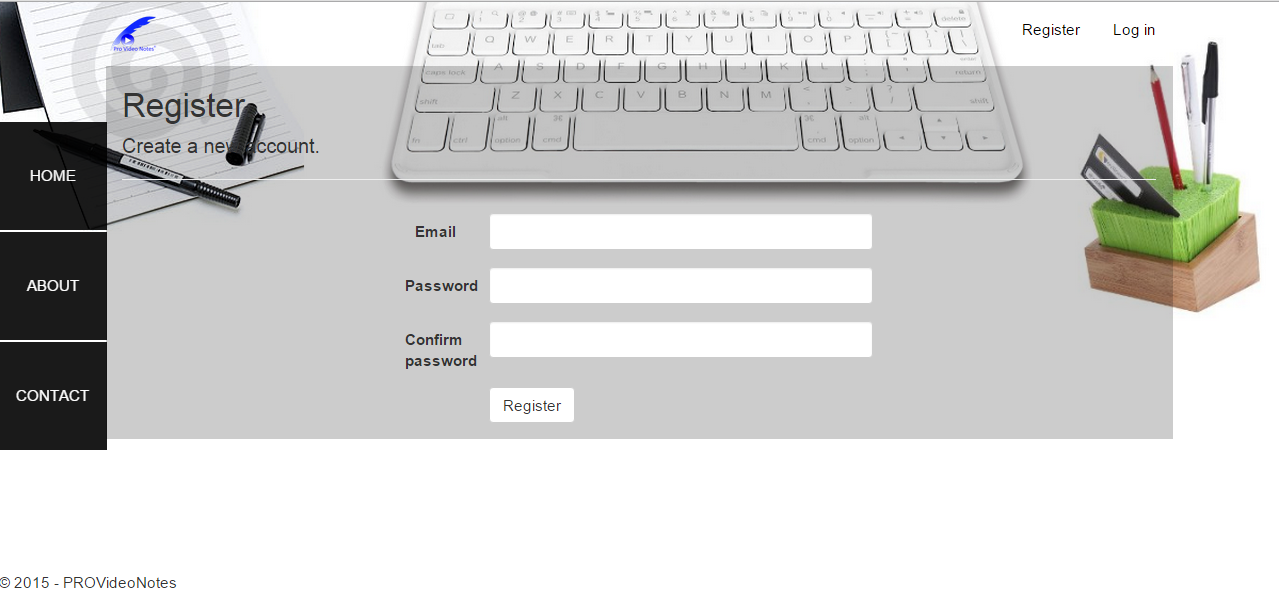
Слика 3.1-2. Почетна страна мобилна верзија

На сликата е претставена мобилната верзија на апликацијата со паѓачко мени за линковите поврзани со најава и регистрација на нов корисник.



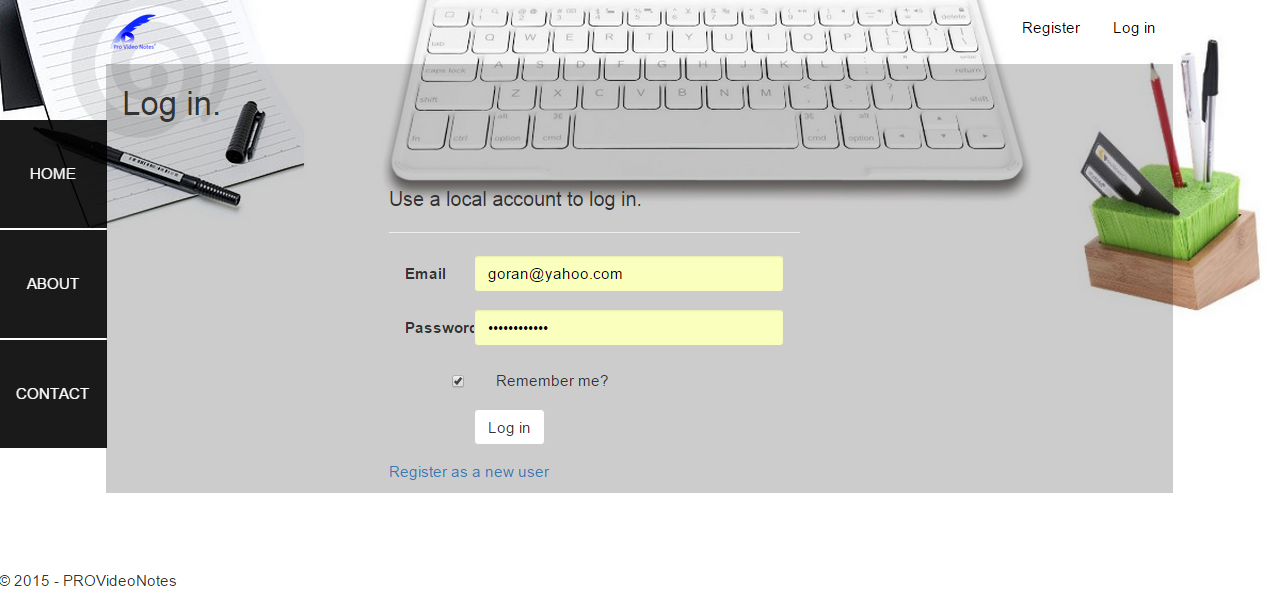
Слика 3.1-3. Почетна страна таблет верзија

## 3.3.Форма за регистрација



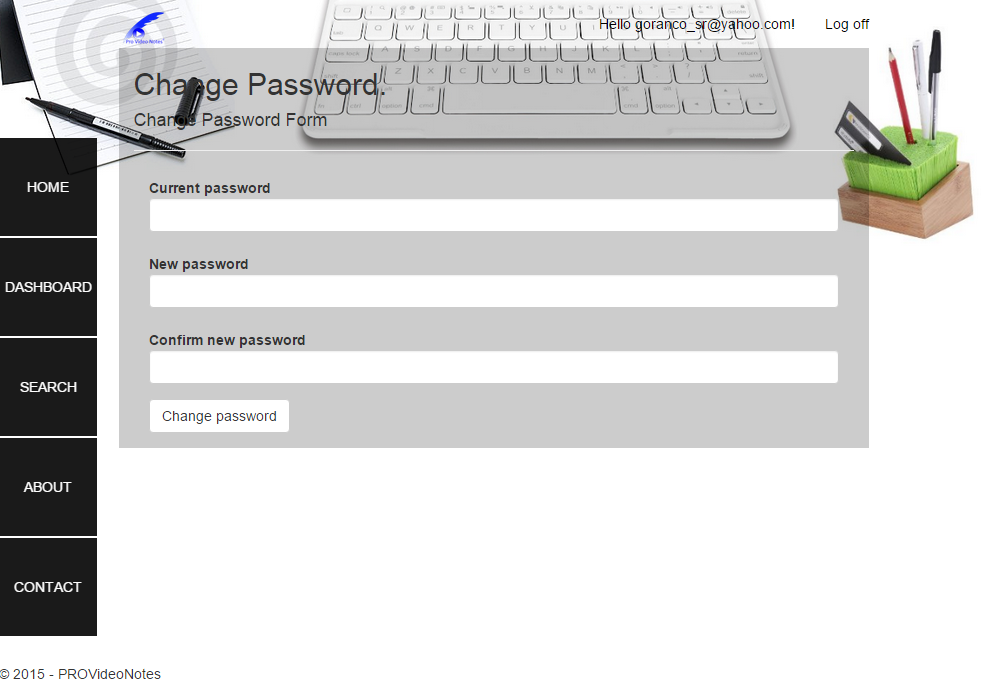
Слика 3.1-4. Форма за регистрација на нов корисник

## 3.4.Форма за најава



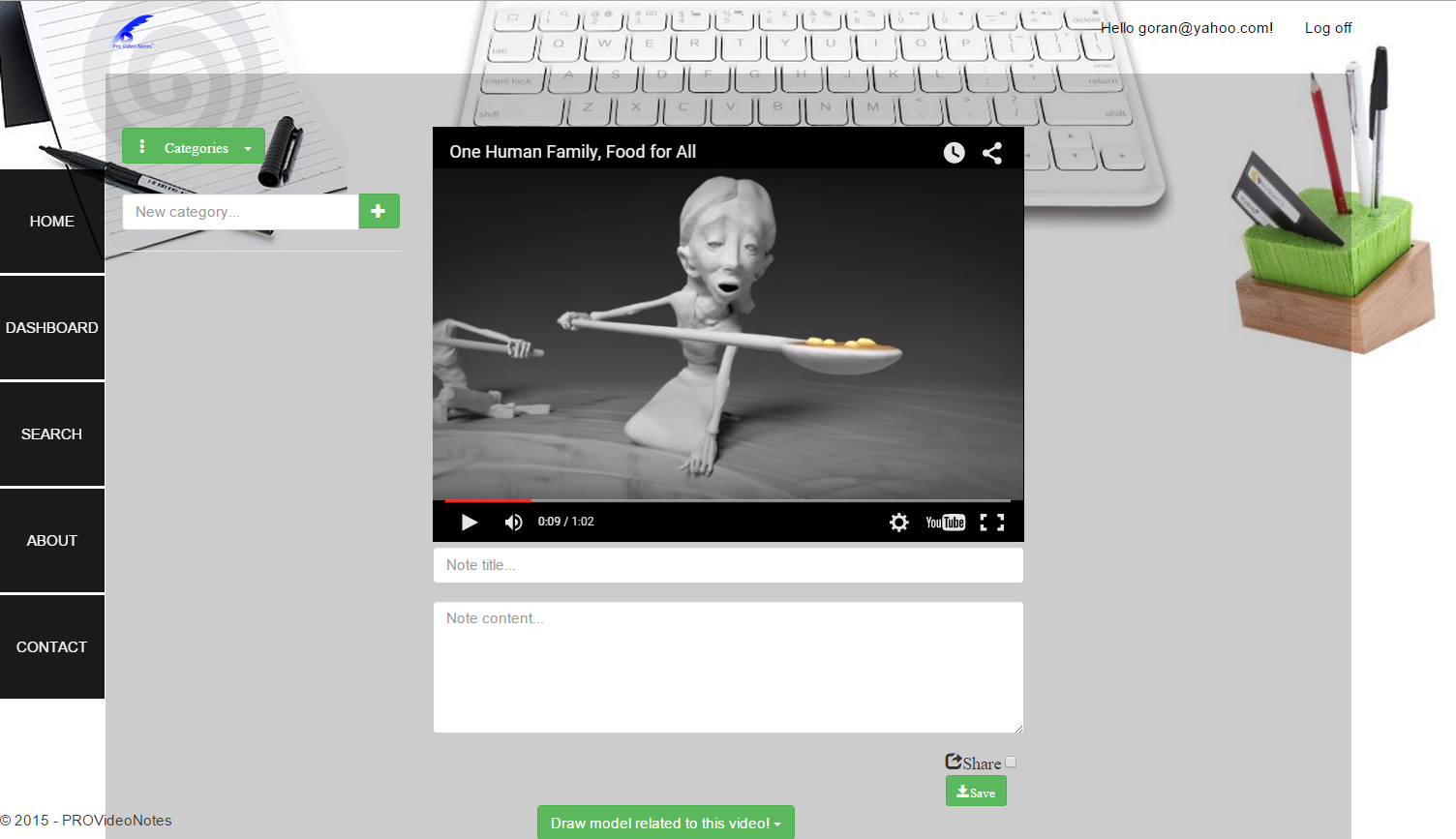
Слика 3.1-5. Форма за најава

## 3.5.Форма за промена на лозинка



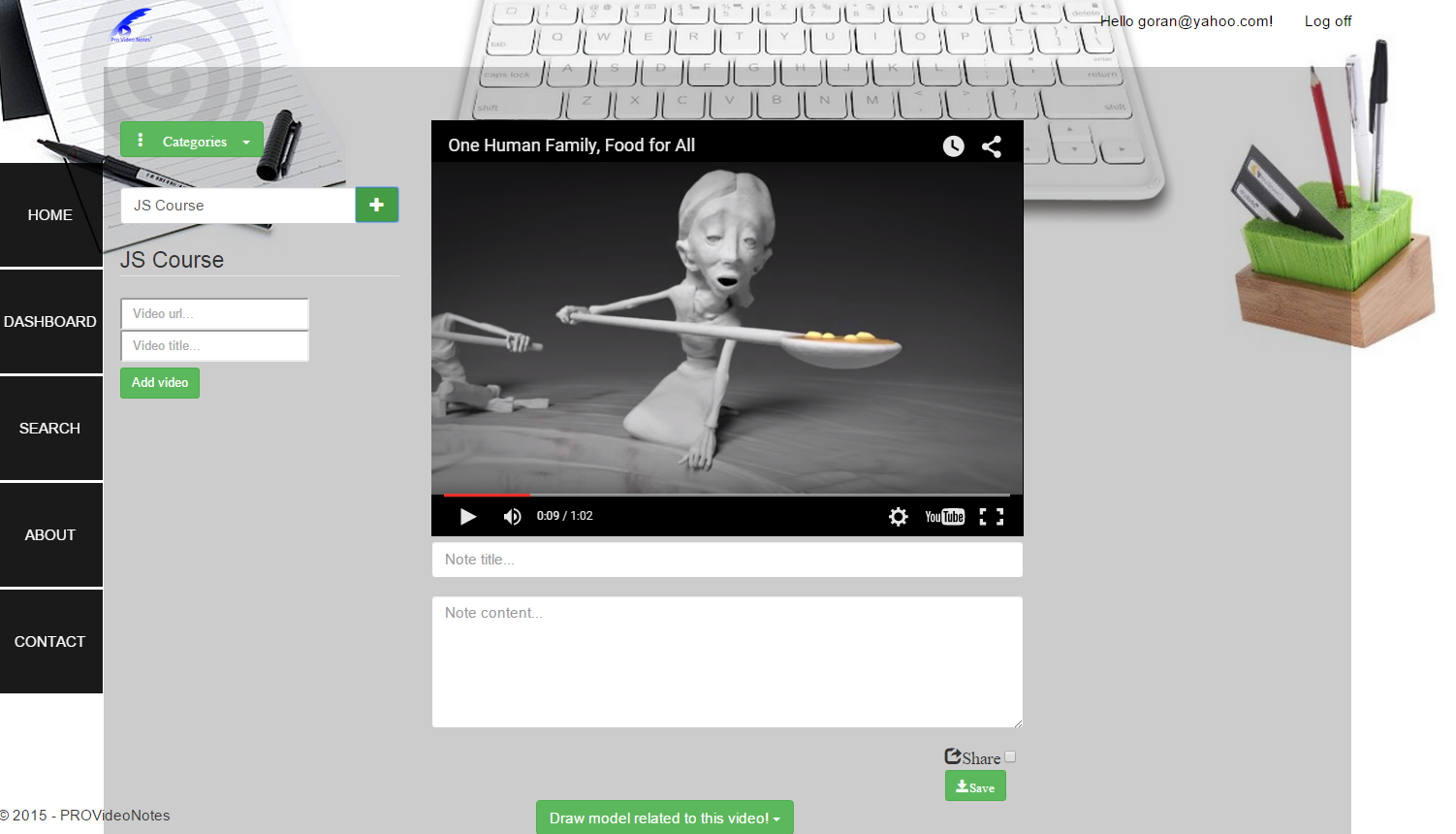
Слика 3.1-6. Форма за промена на лозинка

## 3.6.Dashboard форма за работа со категории, видеа, анотации и 2д графички модели



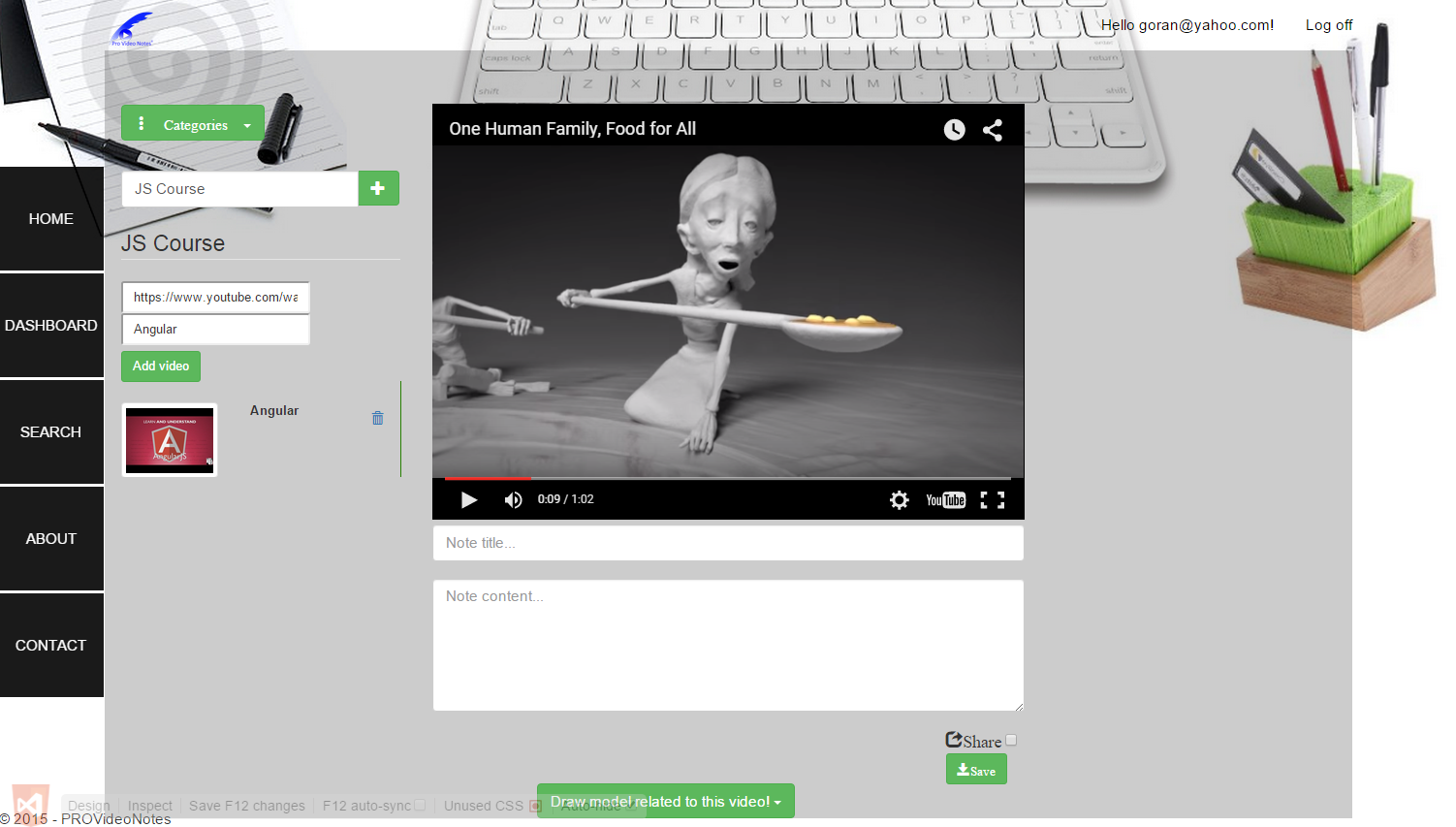
Слика 3.1-7. Dashboard првичен изглед

На сликата е претставен основниот изглед на формата за работа со анотации, категории и видеа. Видеото прикажано на сликата ќе претставува воведно видео на кое ќе се прикаже начинот на работа со самата форма.



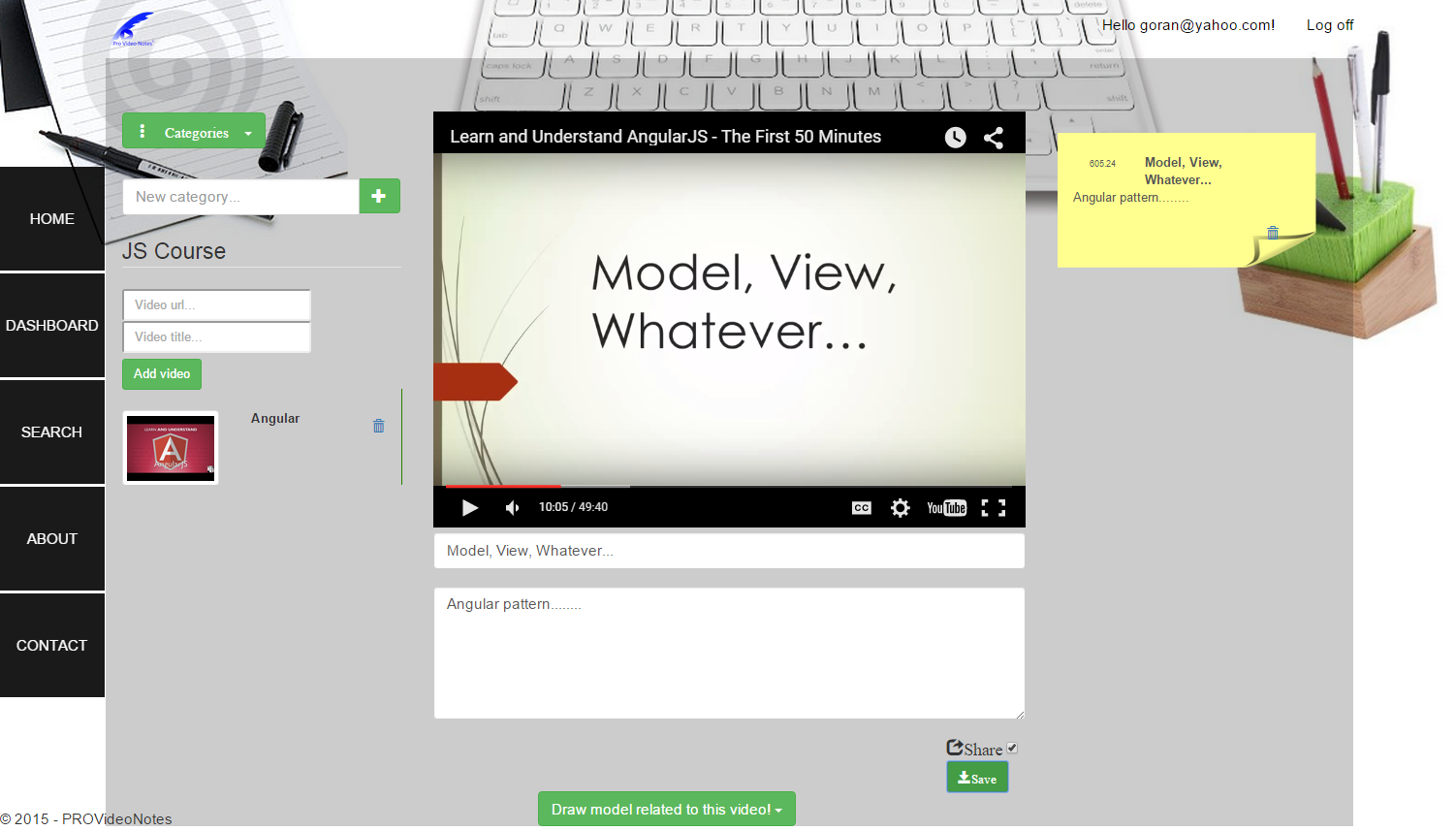
Слика 3.1-8. Dashboard по внесена категорија

По внесување на нова категорија под самото поле се појавува името на категоријата како и полиња за внесување на линк кон видео и именување на истото.



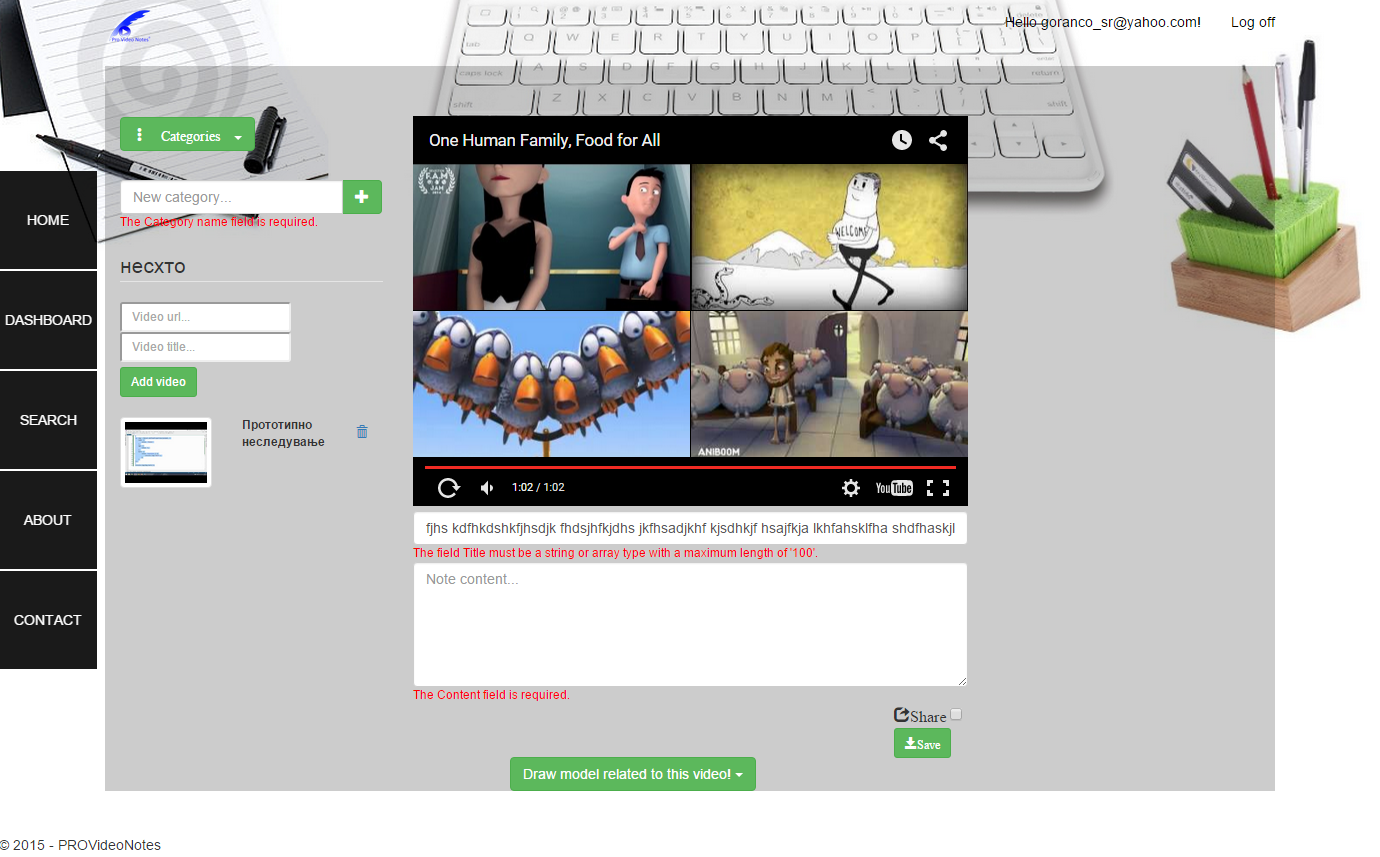
Слика 3.1-9. Dashboard по внесено видео

По внесеното видео истото се прикажува во посебна панела под полињата за внесување на ново видео. Вака внесеното видео претставува дел од избраната категорија и може да се започне со анотирањ на содржината.



Слика 3.1-10. Dashboard по внесена анотација на конкретен сегмент од видеото

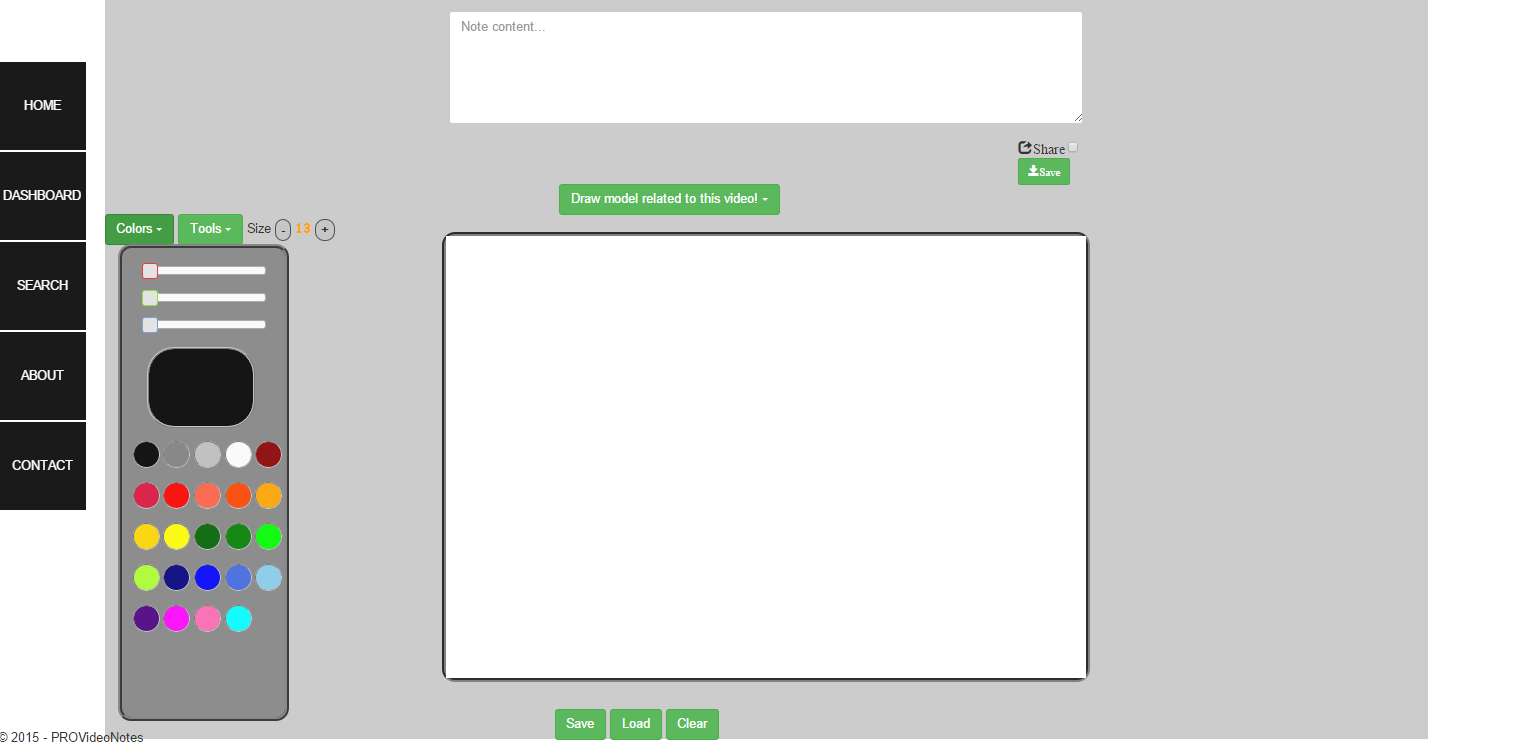
Анотациите се прикажуваат од десната страна и истите имаат можност за бришење. Со клик врз конкретна анотација времето на видеото се мести на сегментот во кој истата анотација била запишана.



Слика 3.1-11. Валидации на клиентска страна

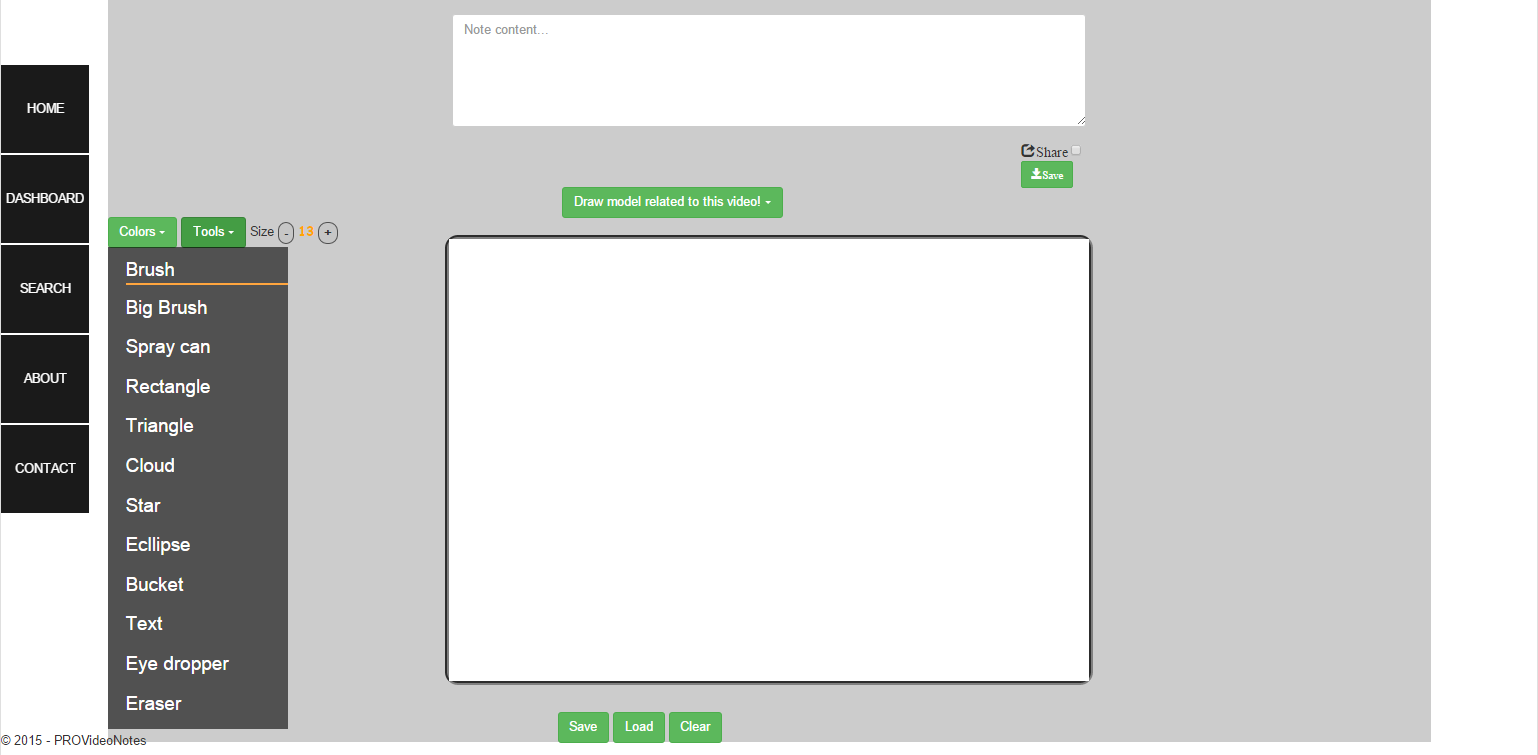
Валидацијата на клиентска страна е овозможена благодарение на DataAnotation атрибутите кои MVC ASP.NET ги нуди.

### 3.6.1.Панела за цртање на 2д модел



Слика 3.1-12. Dashboard панела за цртање на модел поврзан со видеото

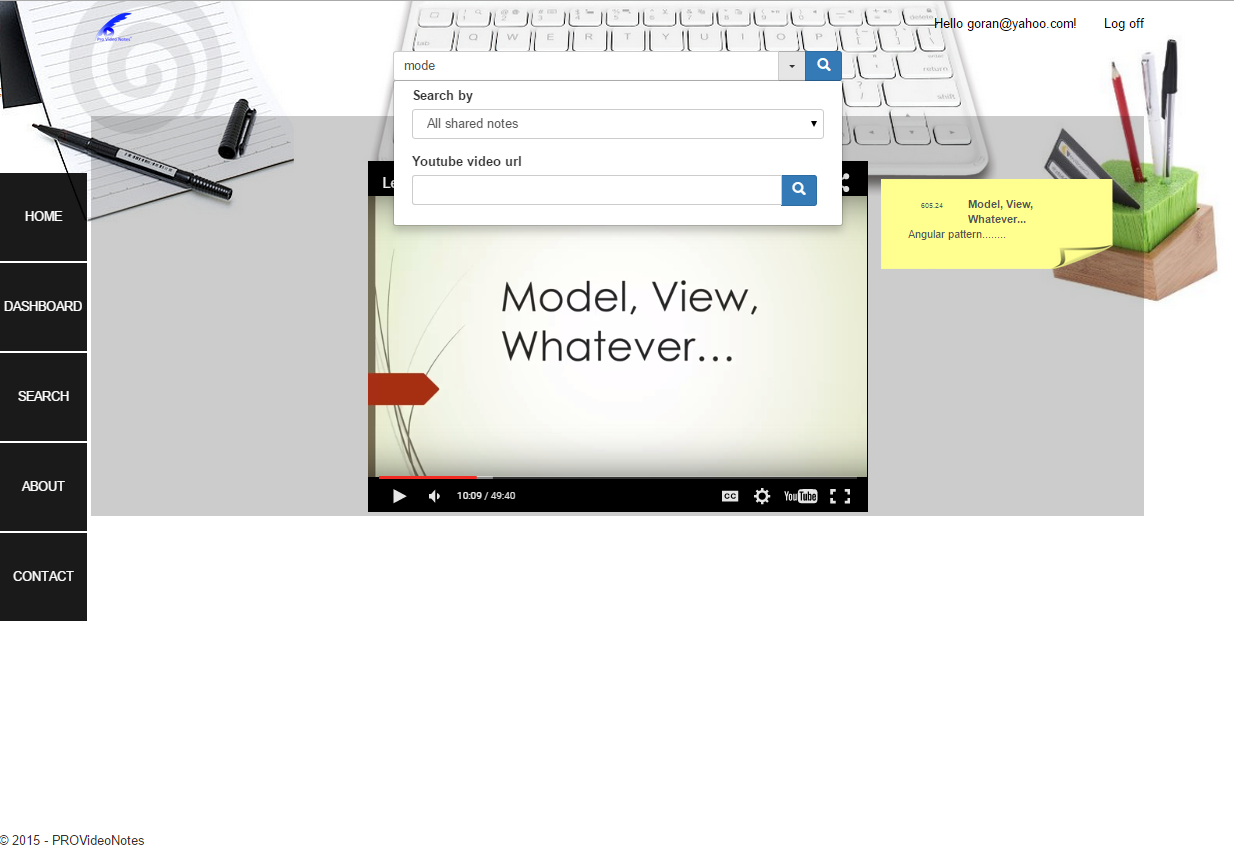
На сликата е прикажана панелата за цртање која овозможува зачувување на моделот во формат на слика на локалната машина.



Слика 3.1-13. Dashboard панела за цртање - готови алатки

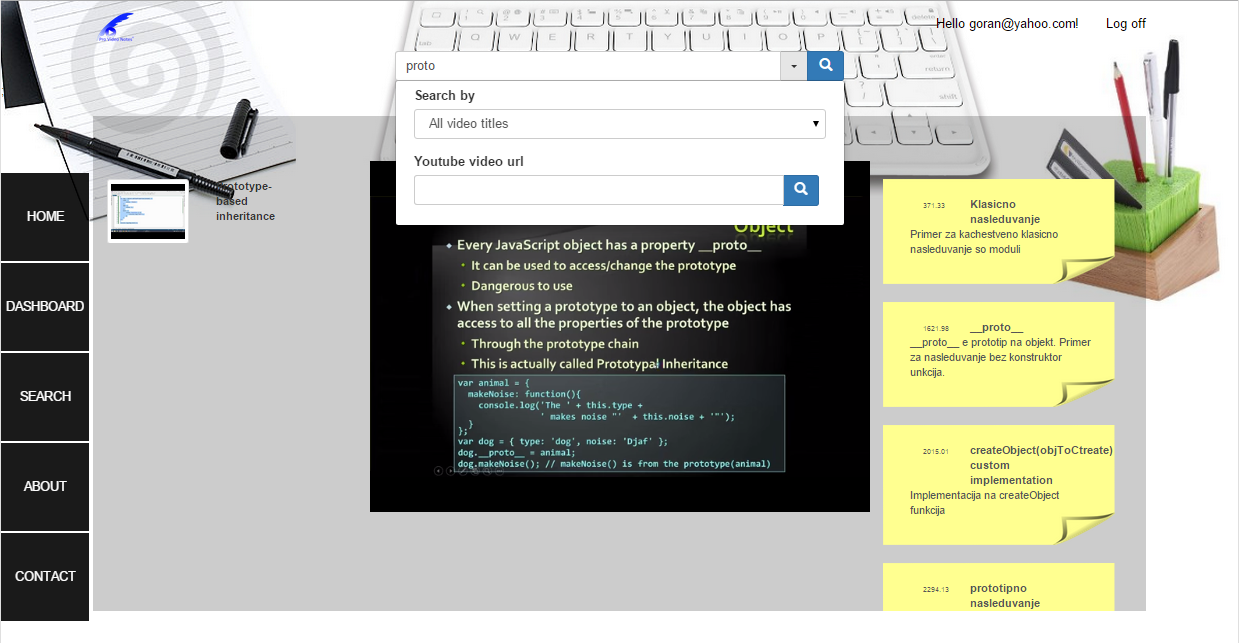
Панелата нуди голем број на алатки кои овозможуваат лесно скицирање на модели поврзани со видео записот.

## 3.7.Search механизам



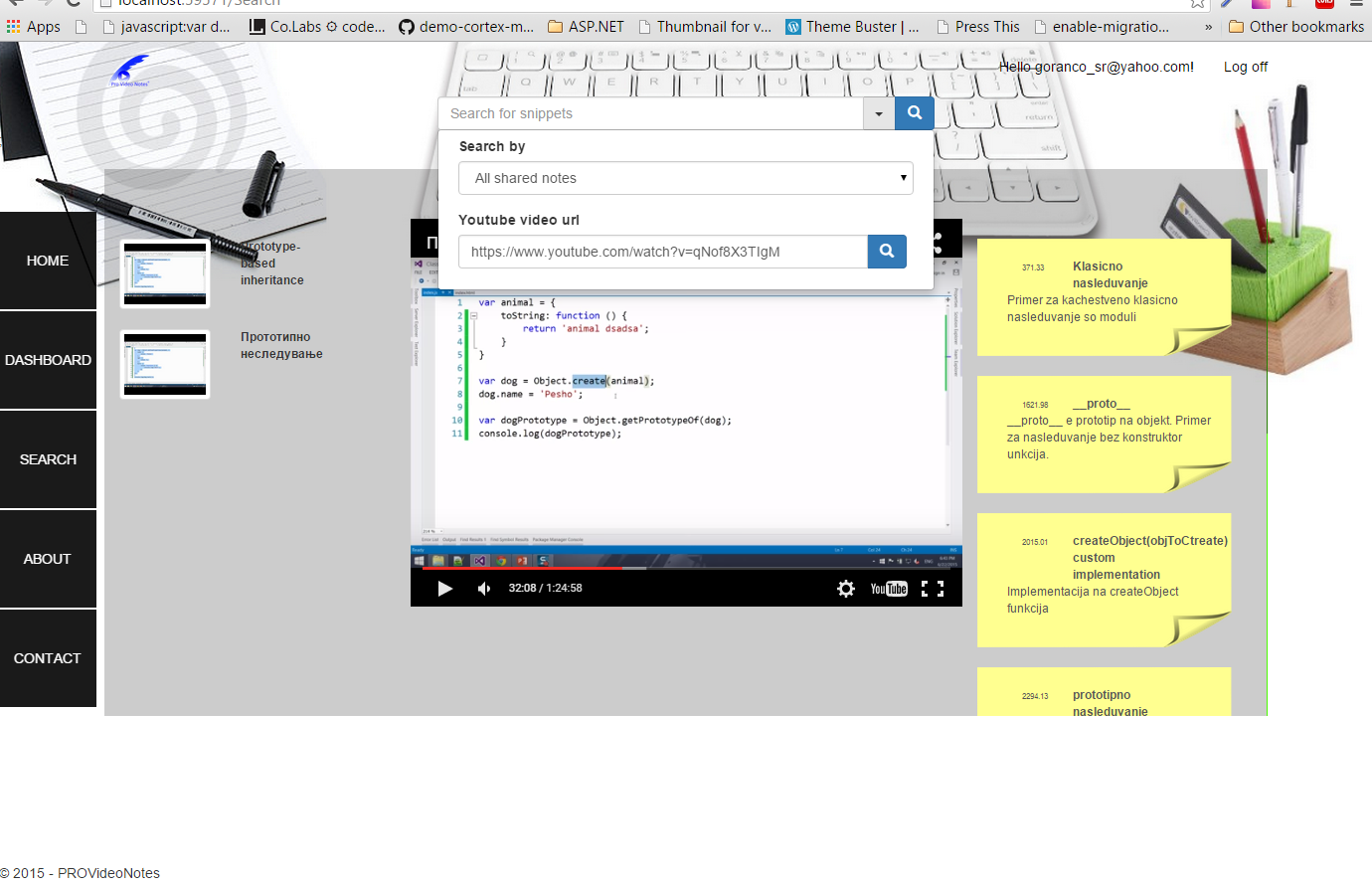
Слика 3.1-14. Страница за пребарување по сите споделени анотации

На сликата е покажана страната за пребарување по сите споделени анотации на корисниците. Пронајдените анотации се прикажуваат од десната страна и со клик врз вив во плеерот се прикажува видеото на конкретниот сегмент на кој кликнатата анотација била запишана.



Слика 3.1-15. Страница за пребарување по наслови низ сите видеа

Резултатите од ваквото пребарување претставуваат видеа кои во насловот го содржат пребаруваниот збор. Со клик врз видеото во плеерот се прикажува истото и сите споделени анотации поврзани со тоа видео.

Слика 3.1-16. Страница за пребарување по youtube линк

Пребарувањето исто така може да се врши и по конкретен youtube линк. Добиените резултати претставуваат лични видеа на корисници чии што заглавја може да бидат на различни јазици соодветно и анотациите кон истите.

# Заклучок

Системот кој е приложен во овој дипломски труд претставува алатка со чија помош, ќе ја зголеми интерактивноста при следењето на онлајн курсеви, со што на корисниците ќе им го олесни едукативниот процес. Со панелата за цртање што овој систем ја нуди, корисниците ќе имат можност да ја искажат својата креативност и преку слика полесно да го запомнат она што го научиле во текот на курсот.

Се надевам дека овој систем ќе биде добро прифатен и истиот ќе стане дел од секој студент кој следи онлајн курсеви. Системот ќе биде поставен како open source на github.

# Референци

1. <https://telerikacademy.com/Courses/Courses/Details/200>
2. <https://msdn.microsoft.com/en-us/data/ff628208.aspx>
3. <https://msdn.microsoft.com/en-us/library/hh404093.aspx?f=255&MSPPError=-2147217396>
4. <http://www.codeproject.com/Articles/838097/CRUD-Operations-Using-the-Generic-Repository-Pat>
5. <http://www.bennadel.com/blog/2379-a-better-understanding-of-mvc-model-view-controller-thanks-to-steven-neiland.htm>
6. <http://stackoverflow.com/questions/20668328/using-asp-net-identity-database-first-approach>
7. <http://stackoverflow.com/questions/14005773/use-asp-net-mvc-validation-with-jquery-ajax>
8. <http://stackoverflow.com/questions/14887871/creating-a-service-layer-for-my-mvc-application>
9. <http://stackoverflow.com/questions/3873530/inject-asp-net-mvc-controller-property-into-service-layer-dependency>
10. <https://en.wikipedia.org/wiki/ASP.NET_Razor_view_engine>