**Assumptions:**

* The developer knows and understands common principles of design of enterprise application ([IoC task](file:///\\proj-datastore\KnowledgeBase\dotNet-internal-tasks\IoC\));
* The developer knows and understands IoC principles ([IoC task](file:///\\proj-datastore\KnowledgeBase\dotNet-internal-tasks\IoC\));
* The developer knows NH or EF ([NHibernate task](file:///\\proj-datastore\KnowledgeBase\dotNet-internal-tasks\NHibernate\) or Entity Framework tasks).

**General notes:**

* The main purpose of this task is deep knowledge of ASP.NET MVC framework. This knowledge is necessary for passing task interview;
* Task interview will be scheduled after all tasks will be completed and reviewed;
* “Need to know/understand something” phrase means that need to be ready to discuss this theme deeply with mentor;
* This version of the document should be copied to the root of your repository branch which will be used for these tasks. If during tasks implementation some mistakes will be found (or you have some recommendations or tasks improvements) then it should be reflected in the mentioned document’s copy. For this action need to enable change tracking in review tab of MS Word;
* In case if some external and useful tasks related resources will be found then it should be reflected in the mentioned document’s copy.

**Documentation notes:**

* ASP.NET MVC books can be found [here](file:///\\proj-datastore\KnowledgeBase\dotNet-internal-tasks\IoC\Books\) . Choose one good book for reading;
* ASP.NET MVC videos can be found [here](file:///\\proj-datastore\KnowledgeBase\dotNet-internal-tasks\IoC\Video\);
* Microsoft recommended ASP.NET MVC resources can be found [here](•%09http:/www.asp.net/mvc/overview/getting-started/recommended-resources-for-mvc).

**Theory:**

* Need to understand what is ViewData, ViewBag, TempData. What's the difference between them?
* Need to know in details and understand ASP.NET MVC application lifecycle;
* Need to understand what is Areas;
* Need to know how to bind complex types, collections ([custom](http://haacked.com/archive/2008/10/23/model-binding-to-a-list.aspx/)/using templates), how to extend binding behavior;

**Implementation notes:**

* All external libraries should be referenced as nugget packages;
* MS SQL database should be used;
* All tasks should be implemented in accordance with [SOLID](http://en.wikipedia.org/wiki/SOLID_%28object-oriented_design%29) principles;
* Hardcode and “magic number” anti-patterns are not allowed;
* Dead code, empty folders, unnecessary commented code are not allowed also;
* The code should be styled in the same manner everywhere;
* The tasks should be implemented using ASP.NET MVC 5;
* Razor view engine should be used;
* ORM should be used for DB access. ORM should be defined by mentor;
* “Need to know/understand something” phrase means that need to be ready to discuss this theme deeply with mentor;
* Need to understand difference between ActionFilters, ResultFilters, ErrorFilters, AuthorizationFilters;
* Need to know all approaches of filters applying.

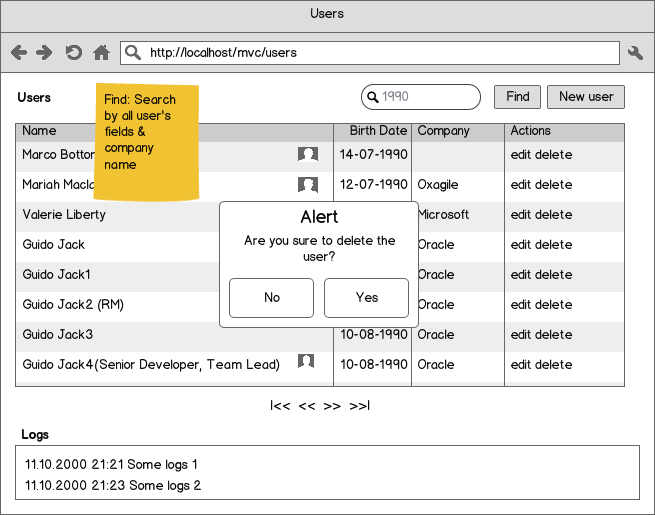


Figure 1.0 “List of users” page

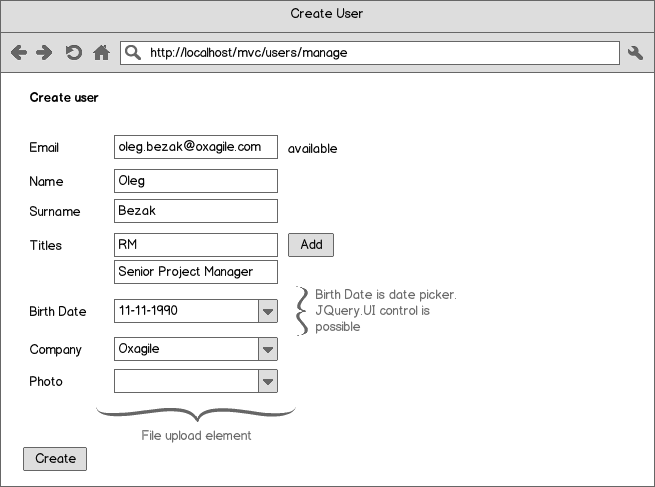


Figure 2.0 “Manage user” page

**Task 1: CRUD**

* Need to implement simple ASP.NET MVC CRUD application. The domain model which covers all requirements of this task displayed on the figure 3.0. Besides this application should meet the requirements below.

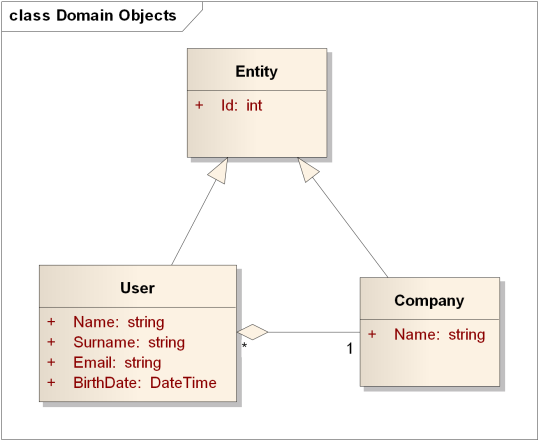


Figure 3.0 CRUD Application domain model

1. The application should contain 2 pages. For more information see figures 1.0 and 2.0;

**Note:** Mentioned figures contain functionality for all tasks which should be implemented outside of current CRUD subtask;

1. “Manage user” page should contain following fields:

* Email;
* Name;
* Surname;
* Birth Date which should be displayed using custom editor template.

No need to implement validation functionality in context of this task.

1. “List of users” page should contain grid with following columns:

* Name which should be build using following template “{Name} {Surname}”;
* Birth Date which should be displayed using custom display template;
* Company;
* Actions.

1. “List of users” should have appropriate functionality for deleting user. See figures for more information.

**Task 2: Validation**

* Need to add data annotation based server validation:
* Email is correct;
* Email is required;
* Name and Surname is required;
* The maximum number of users for one company is N (Configured value).
* The validation should be replaced [to fluent based validation](http://www.nuget.org/packages/FluentValidation/).
* *Need to implement appropriate client based validation (optional).*

**Task 3: Advanced Parameters Binding**

* Need to add “titles” functionality to the application. For more details see figures 1.0, 2.0 and 4.0.

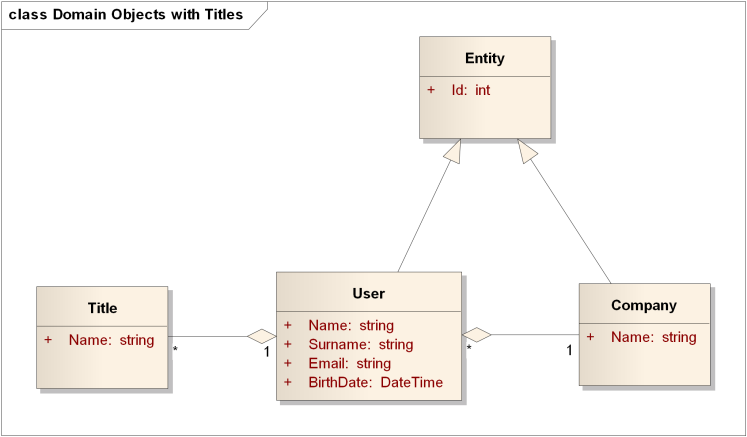


Figure 4.0 Application Domain model with Titles functionality

**Task 4: Filters**

* Need to implement custom filters of all mentioned types. Custom filters should log all method executions for the filters (need to override all possible methods and log all calls). The logs should be displayed for all pages of the application. Concept of master page should be applied. Some details of panned view has displayed on the figure 1.0.

**Task 5: Asynchronous Controllers**

* Add possibility to upload photo for each user. User list should display small version of uploaded photo (icon) for each user.

**Details:**

* Photo resizing should be performed dynamical for each icon request. Asynchronous controllers should be applied in appropriate way.
* DB should contain relative links to the photos only. Blob files should be stored on file system.