The Modules

We separate development into independent modules, and so we divide responsibility. So the main modules are:

1. **Server logic**
   1. **Server core**
   2. **Server business logic**
2. **Authentication logic**
3. **Client business logic**
4. **Google maps logic**
5. **Data Access Layer**
6. **Design**

Server business logic (SBL)

In terms of our proposal, SBL is responsible for implementing queries, getting specific institution details, adding new places, editing existing, submitting the changes (AfU’s function). This module also includes localization, routing and working with Models.

It is a central and the most important module, he plays a role of coordinator between other modules.

Here are the “points of connection” with other modules

1. Client business logic
   1. SBL receives information about the query:
      1. Types of institutions filter
      2. Type of chain(if any)
      3. Name filter
      4. Work time filter
      5. Area filter

This is the query *receive model*

* 1. SBL returns the information about the institutions:
     1. ID
     2. Name
     3. Type
     4. Address
     5. Coordinates
     6. Participation in any chain (and the name of chain)
     7. Icon
     8. Working time(at requested day)
     9. Rating

This is the query *ViewModel*

* 1. SBL receives info about the query for details of the item
     1. Item’s ID
  2. SBL returns info about the details(this is expandable):
     1. Work time for the week
     2. Rating
     3. Last modified by
  3. Receives info about new record(or record edition)
     1. Name
  4. Receives info…

1. DAL – SBL gets all the data from DAL. To abstract away from db, we summarize what information SBL needs from DAL to make some mocks:
   1. List of all existing institutions with their details
   2. The details are
      1. Id
      2. Name
      3. Type (Type ID)
      4. Working time during the week (with possible exceptions, like on holidays etc)
      5. Participation in any chain (chain ID, may be null)
2. Design – SBL’s controllers’ actions should return appropriate View
3. Authentication – SBL depends on it. He expects the following information
   1. whether the user is currently authenticated
   2. his role
   3. his id

Authentication

This module is responsible for determining user’s identity. The basic authentication model will be next :

1. User cannot register his new account
2. In spite of that, he’s offered to log in via known identity providers, such as Facebook, Vkontakte, OpenID, or maybe Gmail(the decision is made to reduce amount of work and confusion with users accounts – change of password, change of name etc)
3. After a login via external system, we receive some kind of his external ID, we call it SID (security ID). If no such token is present in database, we create a new user and set him this token, specifying from which system we received it. If it’s present then we have an internal ID of this user – this is the entity, which the SBL module is interested in.
4. In future we need a functionality to merge different external accounts of the user into one. This is not to be included with basic functionality, but when the system will be working, users will probably ask for it. The only thing we need to know right now is that we will not have many problems implementing such functionality.

Client business logic (CBL)

This module is responsible for the communication with server and manipulating all the data on client side, including transferring data to the view.

CBL is the centralized coordination of any data, available on the client. Here’s how the communication will occur:

1. The SBL gives the information to CBL through JSON in AJAX requests
2. CBL reacts to the receiving of data and updates the javascript collections of data, including ones need to be cached locally for offline work. In this same stage he informs GML to perform drawing and also fills all the Design’s Data.
3. Communication with design occurs through CSS classes
4. Communication with GML occurs through JS classes
5. Communication with SBL occurs
   1. Via Ajax input data on GETs
   2. Via Ajax response JSON’s on POSTs

Google maps logic (GML)

This module is pretty much isolated and dependent only on

CBL. Its only purpose is better visualization of received data (but don’t underestimate the importance of beautiful output)

In future, the level of interaction between CBL and GML ay grow, if we will help the user to get to the desired place by “watching” him (html5 geolocation feature).

Design

Design is as passive as possible and should be developed as an html page at the beginning, later we will merge it with SBL and CBL functionality.