Technical Specification

PhotoContest System

Data Model

- The system holds Users, Contest, Pictures, Prizes, Votes and VotingCommittees.
- **Users** have:
  - Name,
  - Username,
  - Age, (may be better to change this to DateOfBirth)
  - Gender,
  - ProfilePicture,
  - A set of pictures,
  - A set of contests they participate in,
  - A set of contests they have applied to participate in (but still not approved by the contest owner/moderator),
  - A set of contests they have created and are owners/moderators of,
  - A set of votes they have given to different pictures,
  - And a set of VottingCormmittees they participate in.
- **Pictures** have:
  - Author,
  - PictureData - holding the link to the actual picture data kept in a cloud storage provider (e.g. Dropbox),
  - PostedOn,
  - And a set of Contests the picture participates in.
- **Contests** have:
  - Title,
  - Description,
  - StartDate,
  - EndDate,
  - Owner,
  - Status - active, inactive / paused, dismissed, finished,
  - VotingType - open or closed,
  - ParticipationType - open or closed,
  - DeadlineType - when EndDate comes or when a certain number of pictures have participated,
  - A set of prizes,
  - A set of invitees - Users that have applied but have not yet been approved to participate (applicable only to closed ParticipationType contests),

    - A set of participants - Users who have been approved to participate,
    - A set of pictures,
    - A set of votes,
    - And a Committee (Jury) - applicable for contests with closed VotingType.
- **Prizes** have:
    - Name,
    - Description,
    - Contest,
    - Picture - to be decided.
- **Votes** have:
    - Voter,
    - Contest,
    - And Picture
- **VotingCommittees** have:
    - Contest,
    - And Members

Controllers, Actions and Views

1. AccountController

account/register - GET and POST

- POST action accepts **RegisterUserBindingModel** - Name, Username and Password.
    - Allows short and simple passwords, e.g. 123

account/login - GET and POST

- POST action accepts **LoginUserBindingModel** - Username, Password.

account/logout - POST

- Redirects to Home, Index

account/changepassword - GET and POST

- POST action accepts **ChangePasswordBindingModel** - CurrentPassword, NewPassword, ConfirmPassword.

2. HomeController

home/index/{sortBy} - GET

Anonymous users can browse the full list of photo contests. Using pagination. There are options to sort the contests based on popularity, recency and state (active, finished, past contests). The are a links for registration and login.

- The returned View expects model of type **IEnumerable<SummaryContestViewModel>**.

- The View uses **ContestsSummariesPartial** - visualisation of a set of contests.
    - ContestsSummariesPartial contains ContestSummaryPartial - one for each contest in the set.
- Uses Ajax requests to change the order for displaying the contests.

3. MeController

Requires Authorisation.

me/index/{sortBy} - GET

Logged-in users have access to similar information as anonymous users and in addition ability to create and manage their own contests, to upload their pictures, to participate in contests and to vote.

- The returned View expects model of type **IEnumerable<SummaryContestViewModel>**.
- The View uses **ContestsSummariesPartial**.
    - ContestsSummariesPartial contains **ContestSummaryPartial**.
- Uses Ajax requests to change the order for displaying the contests.

me/contests - GET

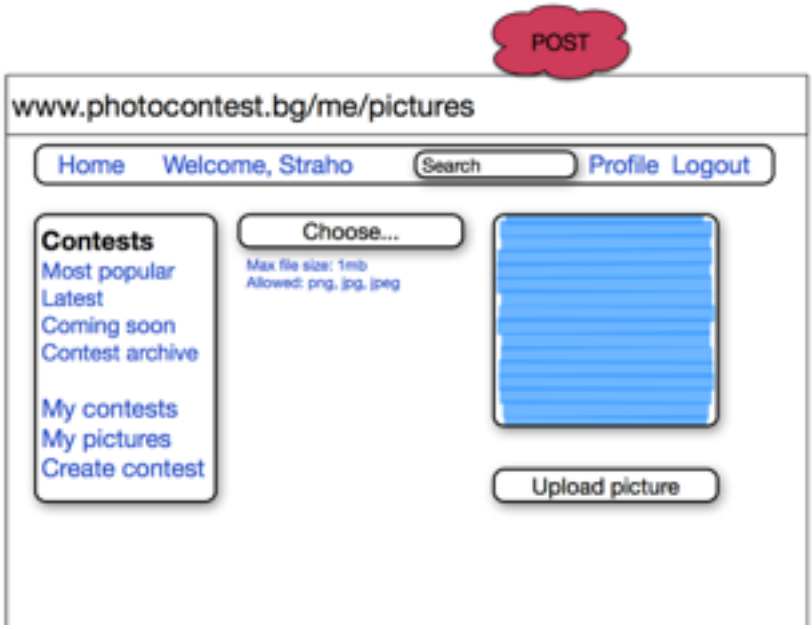
Displays a list of the contests where the current user is the author / moderator.

- The returned View expects model of type **IEnumerable<SummaryContestViewModel>**.
- The View uses **ContestsSummariesPartial**.
    - ContestsSummariesPartial contains ContestSummaryPartial.

me/pictures - GET and POST (for picture upload)

Displays a list of the pictures uploaded by the User.

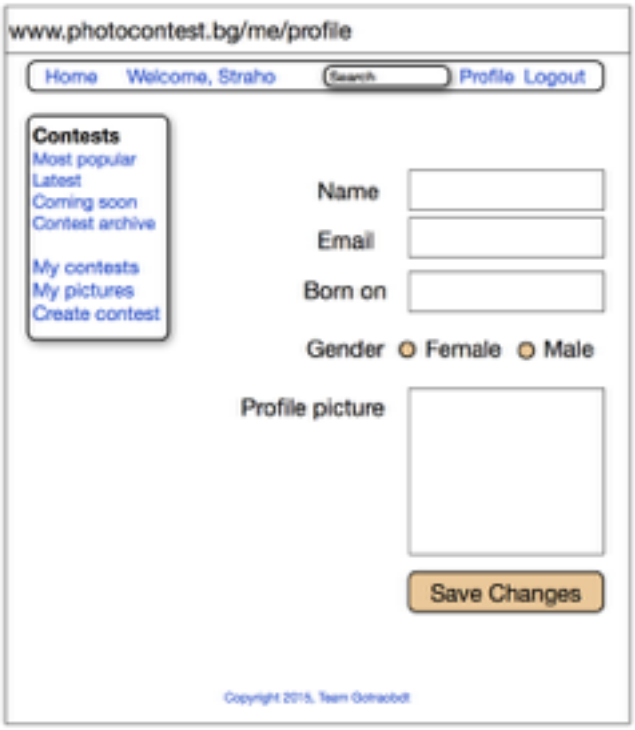
- The returned View expects model of type **IEnumerable<SummaryPictureViewModel> -** summary info about a picture.
- The View uses **PicturesSummariesPartial**.
    - **PicturesSummariesPartial** contains **PictureSummaryPartial -** visualisation of a single photo in a thumbnail form.



me/profile - GET and POST

Displays a form to the User to change his/her profile information: Name, Gender, ProfilePicture, BirthDate.

- Uses **EditProfileBindingModel**.



4. ContestsController

contests/create - GET and POST

Requires Authorisation.

Displays a form to the User for creating new contest.

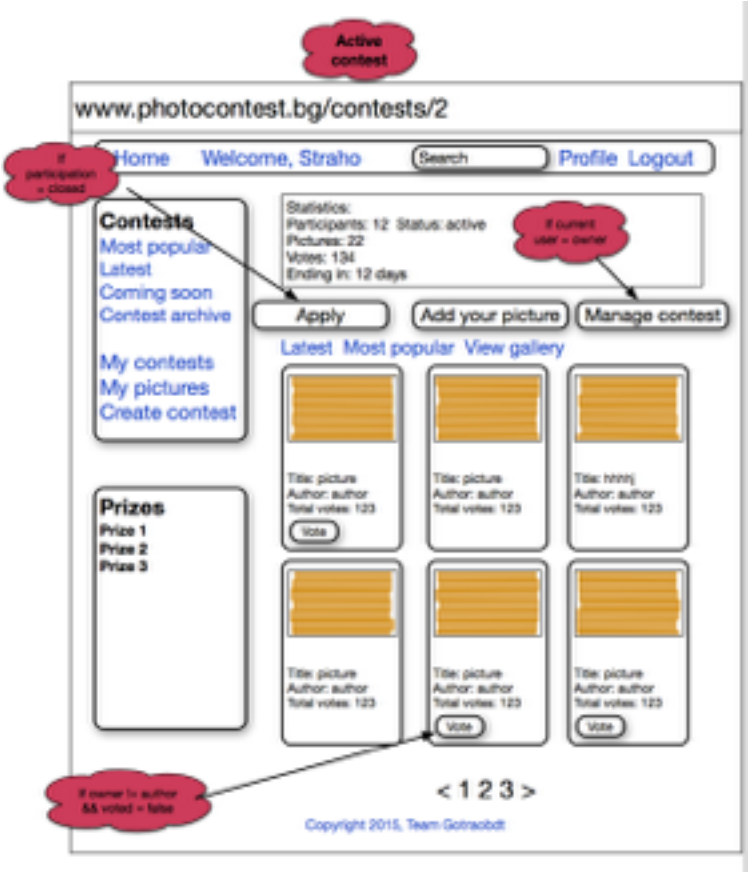
- POST action accepts **CreateContestBindingModel**.

contests/{contestId} - GET

Displays summary info about the contest and a list of participating pictures (using pagination). Displays list of prizes the contest winners will get.

Gives option to sort the pictures by recency and by popularity.

Has link to the contest gallery.



If the contest is in finalized state - displays on top the winners and the winning pictures.

Authorized users can vote for pictures (only one vote per picture) if the contest is open for voting. If it is closed - they can vote only if they are members of the jury. Votes are allowed only for contests with status == Active.

They can participate with pictures if the contest is open for all users. If it is closed they need to apply. When they are approved by the contest owner - then they can participate.

Participating is allowed only for contests with status == Active.

Contest owners can manage the contest (They see button 'Manage Contest').

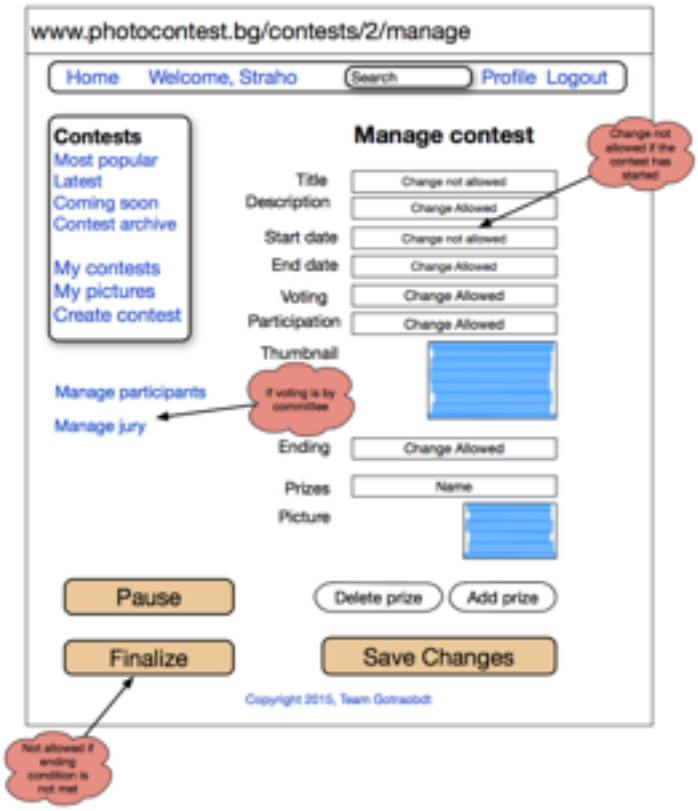
- The View accepts **DetailsContestViewModel** and **IEnumerable**<**SummaryPictureViewModel**>.
- Uses Ajax requests to change the order for displaying the pictures - latest, most popular.
- Uses SignalR for real-time updates of the number of votes the pictures have.

contests/{contestId}/manage - GET and POST

Requires Authorisation.

Accessible only to contest owner. If current user != contest owner returns Not Authorised.

Displays a form to the User where he/she can change important contest parameters such as description, start date and end date. Title is not allowed to change.



StartDate can be changed if the contest has not already started.

If voting type is changed from open to closed all votes given until now must be deleted.

If participation type is changed from open to closed all users already in with pictures are allowed to continue.

If the contest is closed for voting we display a link for managing the jury.

We display a link for managing the participants.

We display button for pausing the contest - only if the contest has started. The contest state is changed to Inactive - no voting and participation is accepted during this state.

If contest ending condition is met (we have reached the end date or the needed number of participating pictures is reached) we display a Finalise button. It changes contest state to Finalized and redirects to the contest page - this time showing the winners and winning pictures.

- Uses **EditContestBindingModel**.

contests/{contestId}/jury - GET and POST

Requires Authorisation.

Accessible only to contest owner. If current user != contest owner returns Not Authorised.

Accessible only if the contest has closed voting type. If contest is open for voting returns Not Found.

Displays a list of the jury members. Displays buttons to remove a member and to invite another member.

- Uses Ajax requests to remove members from the jury.

contests/{contestId}/candidates - GET and POST

Requires Authorisation.

Accessible only to contest owner. If current user != contest owner returns Not Authorised.

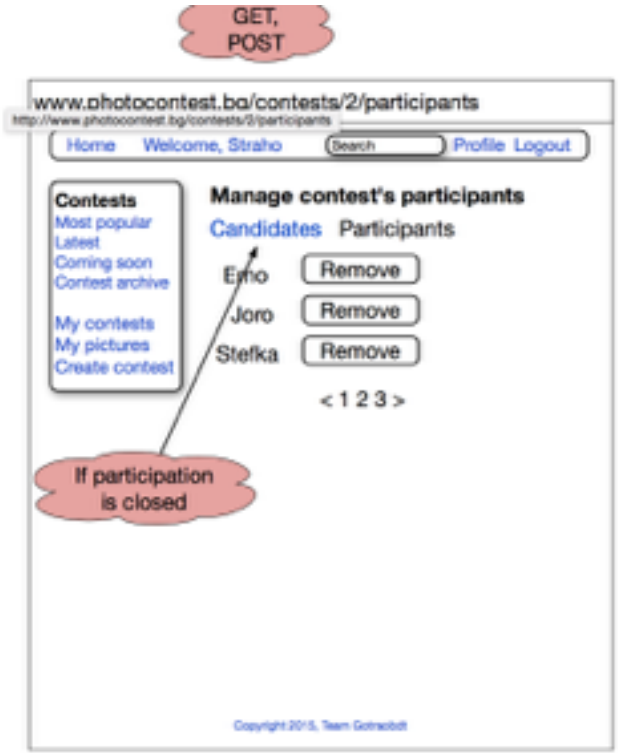
Accessible only if the contest has closed participation type. If contest is open for participation returns Not Found.

Displays a list of the users applied for participation. Displays buttons to approve or reject them.

- Uses Ajax requests to approve or reject users.
- Uses pagination to display the users.

contests/{contestId}/participants - GET and POST

Requires Authorisation.



Accessible only to contest owner. If current user != contest owner returns Not Authorised.

Displays a list of the participating users. Displays button to remove a given user - for example if the user uploads inappropriate content and/or violates contest rules.

- Uses Ajax requests to remove users.
- Uses pagination to display the users.

contests/{contestId}/gallery/{pictureId} - GET

Displays all pictures from the contest, one by one, with arrows to move to next / previous image. Displays button for voting if user is logged and has not voted already.

- Uses Ajax requests to submit votes.
- Uses SignalR for real-time updates of the number of votes the pictures have.
- Uses **DetailsPictureViewModel.**

contests/{contestId}/vote/{pictureId} - POST

Requires Authorisation.

Adds a vote to the specified picture for the specified contest from the currently logged-in user via Ajax request.

Returns the updated votes count.

Uses SignalR to update in real-time the votes count for all other users.

5. PicturesController

pictures/{pictureId} - GET

Displays a picture and detailed info about it.

If user is logged-in and user == owner buttons for deleting the picture and editing the picture are displayed.

- Uses **DetailsPictureViewModel.**