**Topic of the project: “The Maze” game**

**Annotation:** We build a maze game, where the user has to move from start to finish through the walls without touching them as fast as possible. The database is created for keeping information about games history and results, which can be shared to social network using API.

**Members of the team and their roles:**

Osmanova Yana 146 - GUI, application logic  
Badaeva Elina 146 - DB, application logic

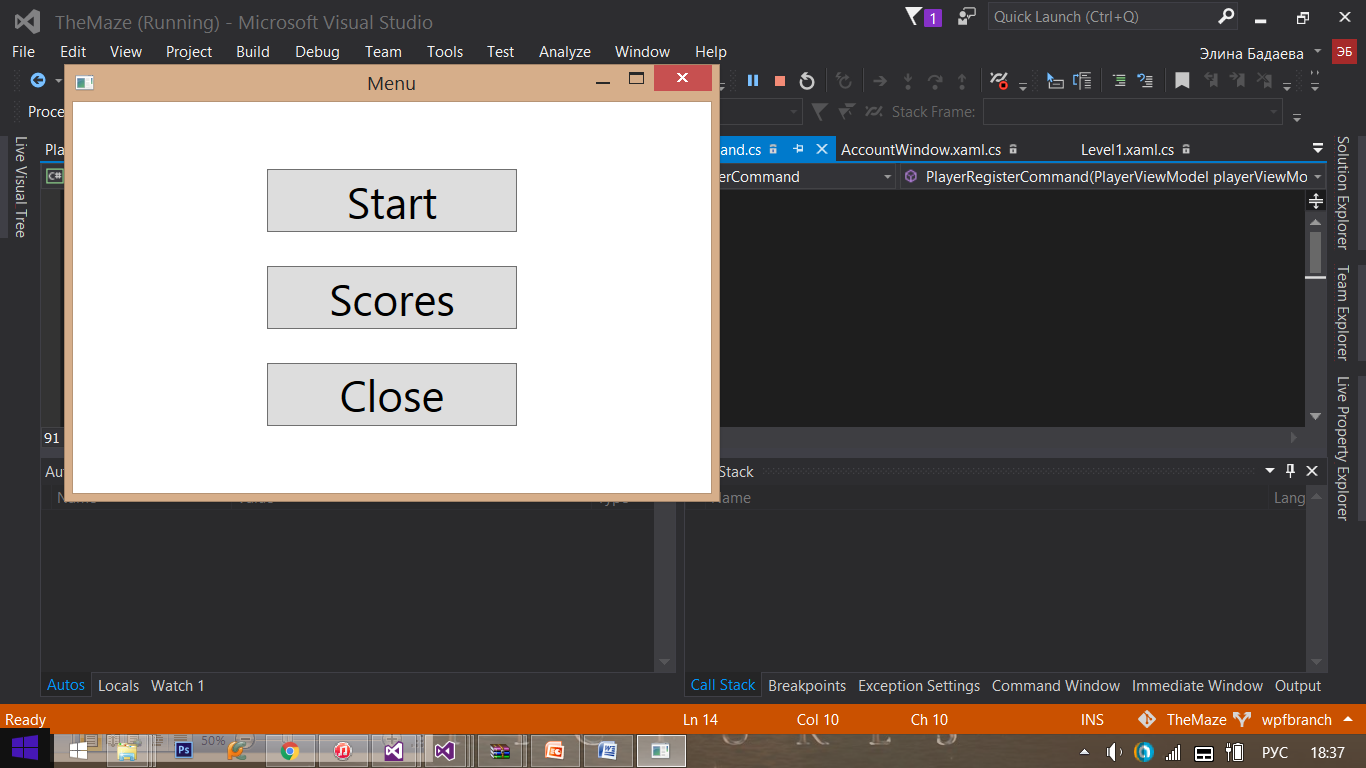
**Central repository address:** <https://github.com/d1stort/TheMaze.git>

**List of classes:**

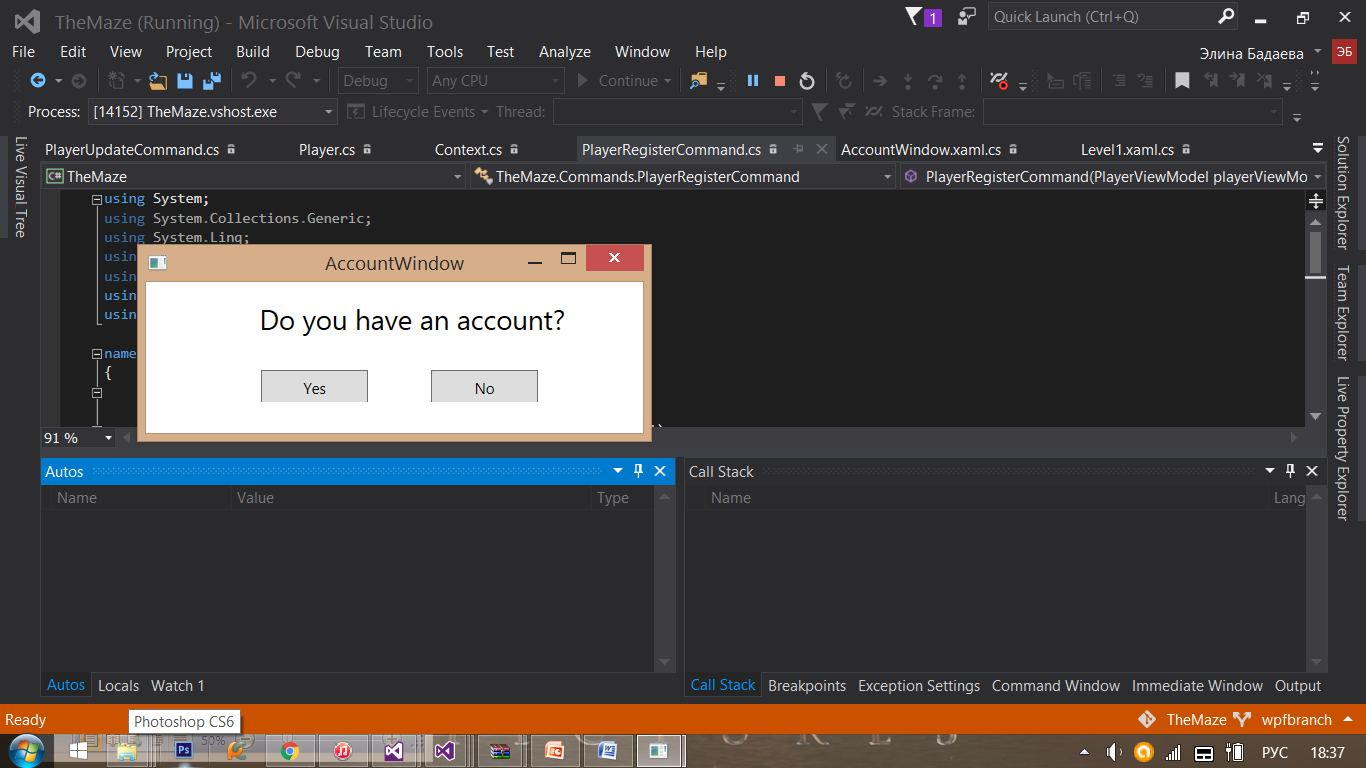
1. public partial class MainWindow - (private void closeButton\_Clicked, private void startButton\_Clicked, private void statsButton\_Clicked, static IQueryable<Player> LoadPlayersFromDB, private void ShowPlayers, private void goBackButton\_Clicked) allows to control the main logic of application. For instance, it has methods for handling clicked button events.
2. class PlayerViewModel - is the bridge between the view and the model. The ViewModel retrieves data from the Model and manipulates it into the format required by the View. It notifies the View if the underlying data in the model is changed, and it updates the data in the Model in response to UI events from the View.
3. public partial class RegistrationWindow : Window - (private void CloseRegWindButton\_Click, public RegistrationWindow()) allows to register your nickname in case you have not already have an account.
4. public partial class LoginWindow : Window - (private void PlayButtonLW\_Click, public LoginWindow()) allows to log in to the application for keeping player’s game history.
5. public partial class Level1 : Window - (private static extern bool SetCursorPo, private static void SetCursor, public Level1(), private void mouseOver\_finish, private void mouseEnter\_label, private void Level\_Loaded, private void BackTMenu\_Click) allows to move the mouse pointer from the start to the finish, touching any of the walls is banned.
6. public partial class AccountWindow : Window - (public AccountWindow(),private void YesButtonAc\_Click, private void NoButtonAc\_Click) allows to understand whether you have an account or not. If you have, the LoginWindow will be opened. Otherwise, the RegistrationWindow will be opened.
7. public class Player - (private void OnPropertyChanged) is initial domain class with entities. It provides a view-independent representation of entities. The design of the model is optimized for the logical relationships and operations between entities.
8. class Context: DbContext - (public Context() : base("PlayersDb"), public DbSet<Player> Players, protected override void OnModelCreating)
9. internal sealed class Configuration : DbMigrationsConfiguration<TheMaze.Context> - (public Configuration(), protected override void Seed) derives from DBContext class and exposes DbSet properties.
10. internal class PlayerUpdateCommand : ICommand - (public PlayerUpdateCommand, public bool CanExecute, public event EventHandler CanExecuteChanged, public void Execute) includes methods for turning the LoginButton active and for processing activities in windows.
11. internal class PlayerRegisterCommand : ICommand - (public PlayerRegisterCommand, public bool CanExecute, public event EventHandler CanExecuteChanged, public void Execute) includes methods for turning the RegisterButton active and for processing activities in windows.

**Program interface:**

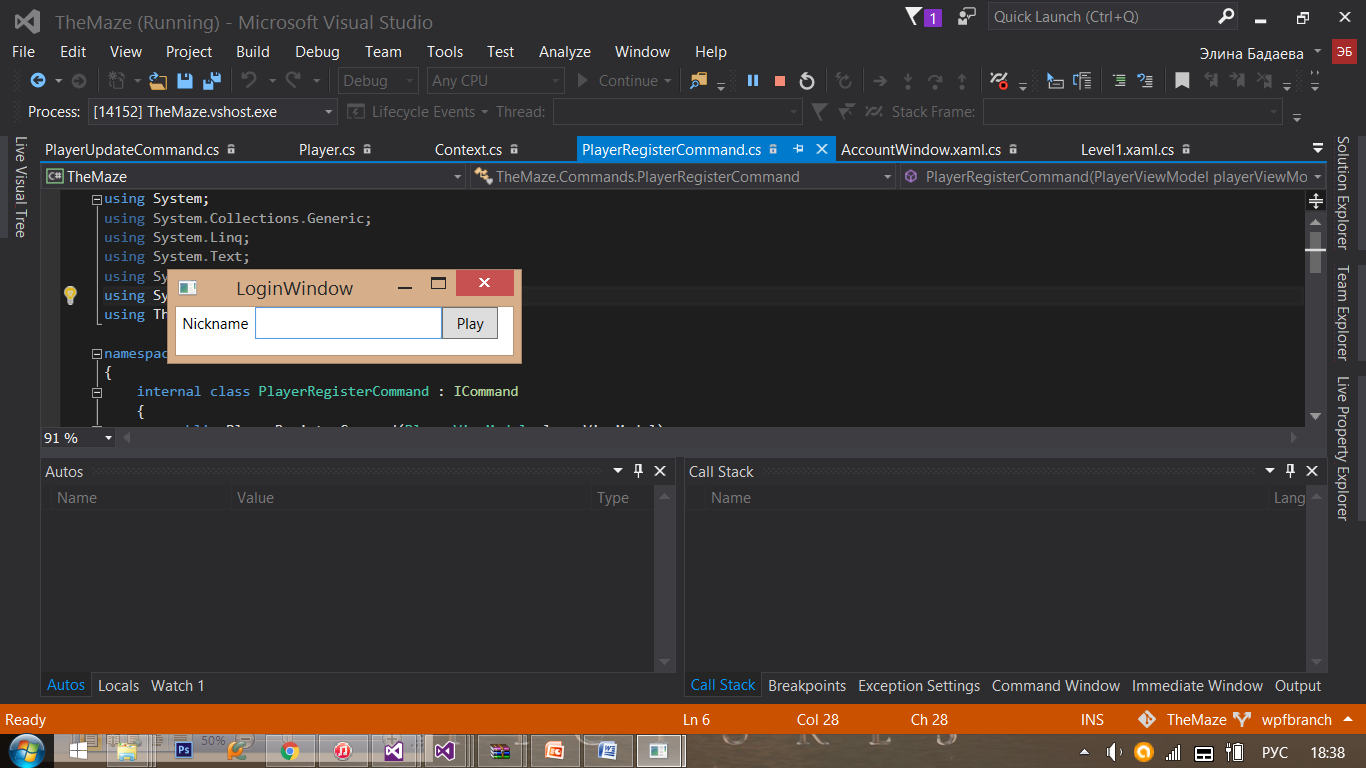
MainWindow allows selecting one of three in the menu: Start, Scores, Close

****

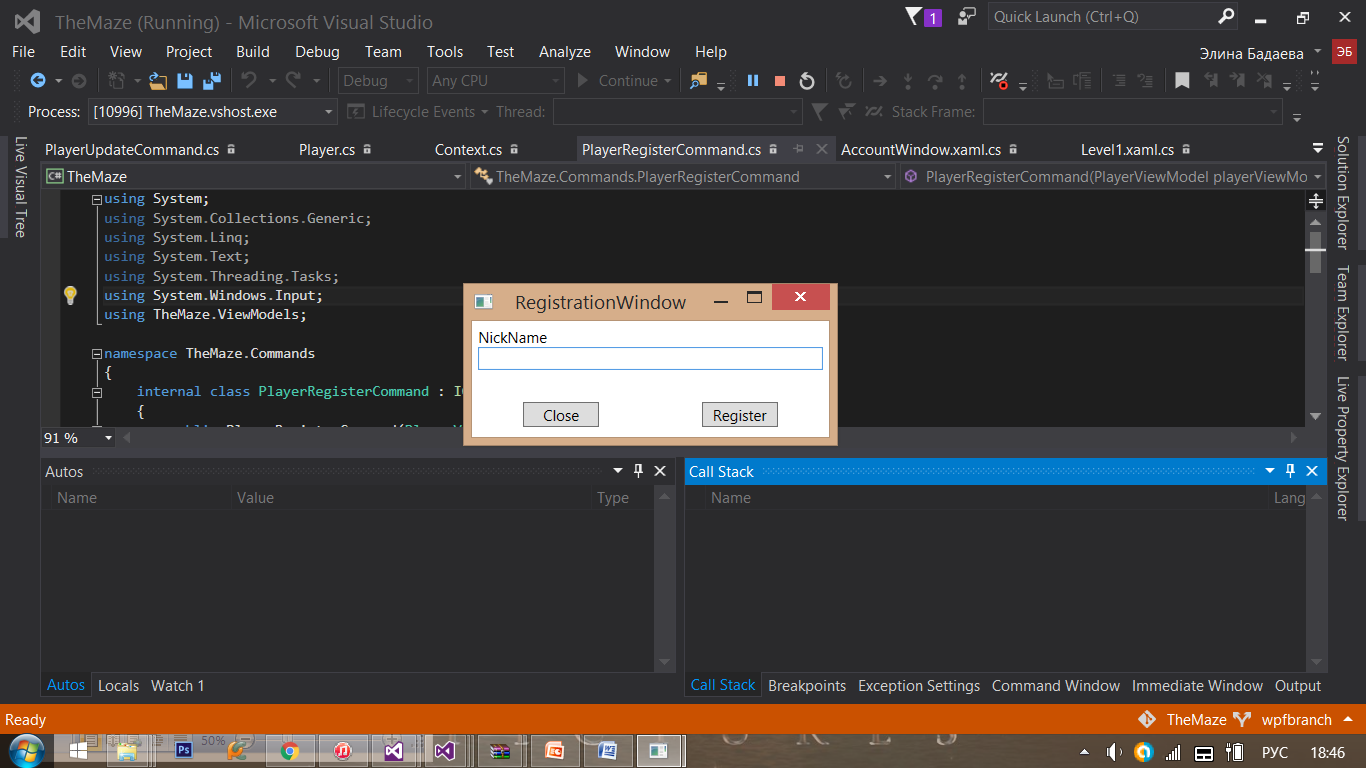
After selecting Start-button you are asked whether you have an account or not.

****

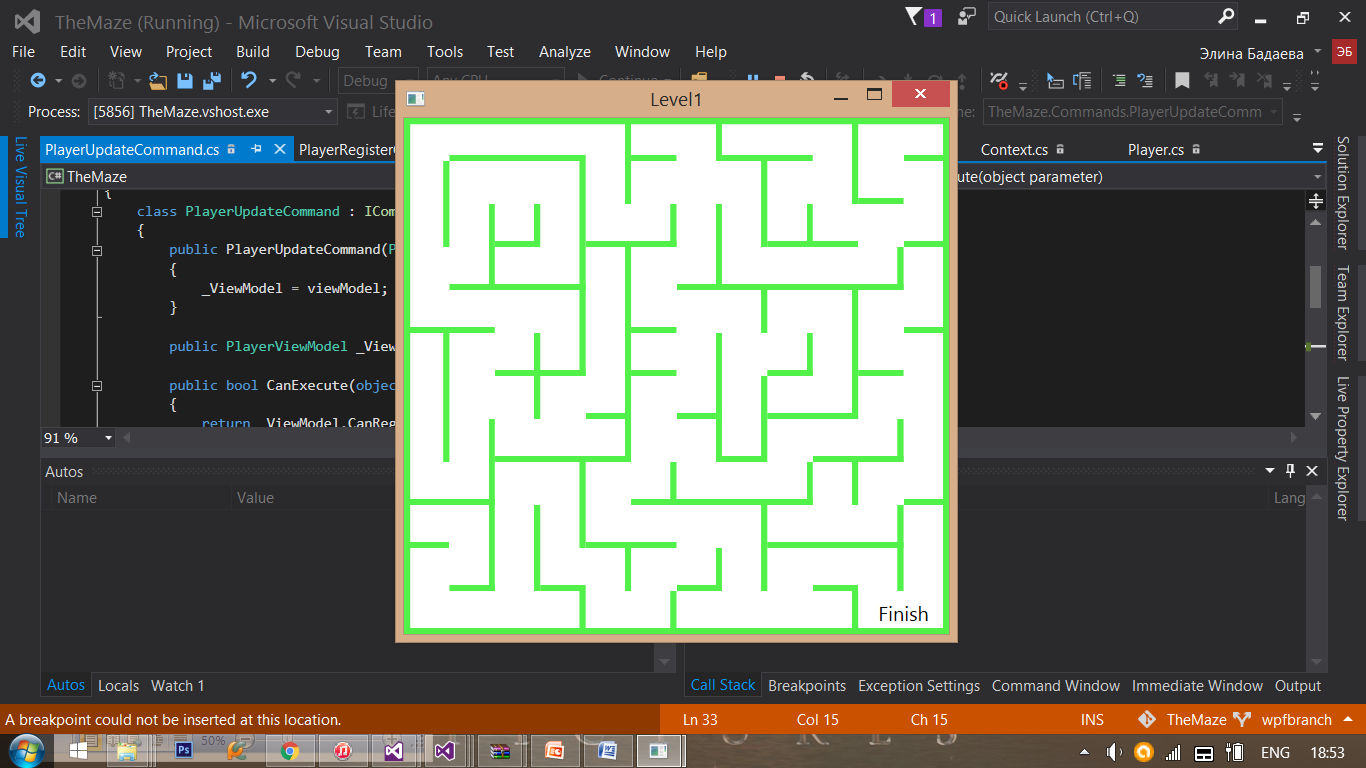
If yes:

****

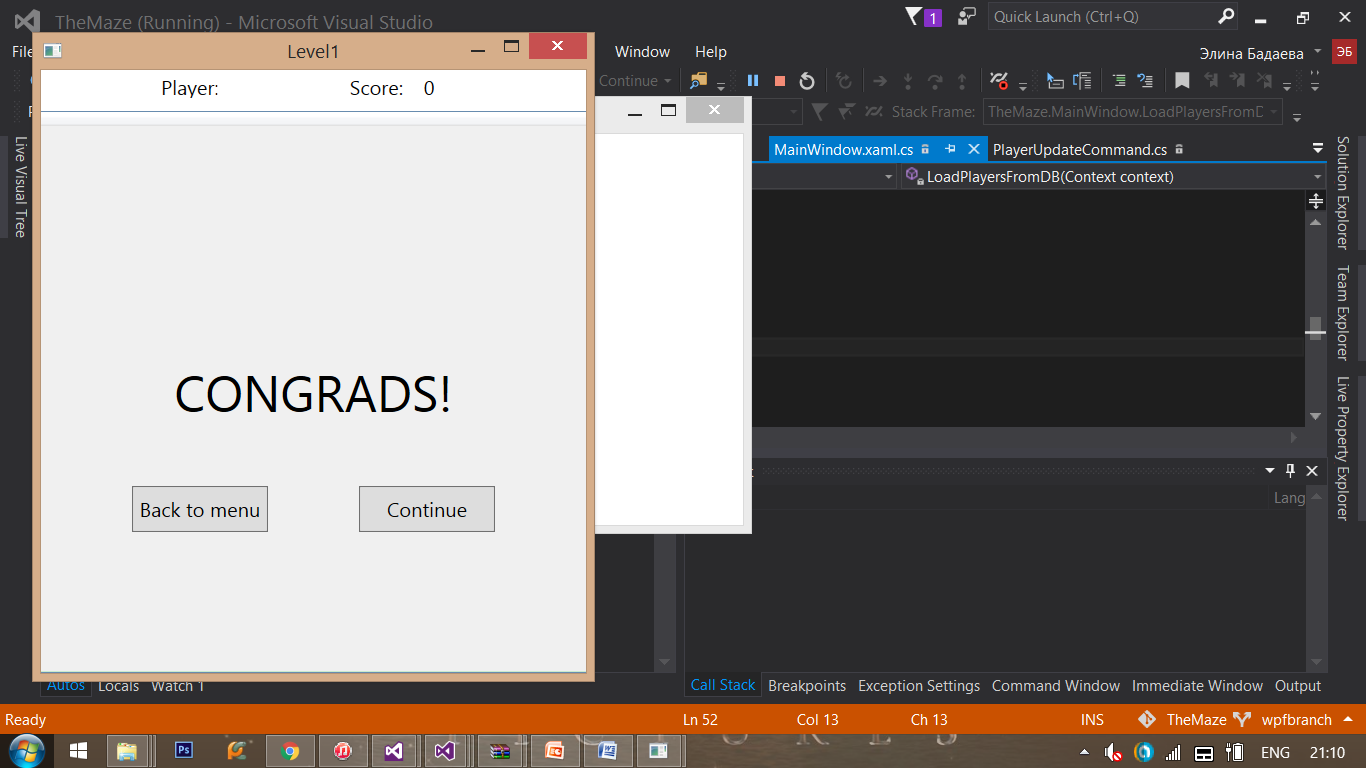
If no:

****

After successful logging in or registration you are suggested to try the first Level of the game.

****

If you win, you will get congratulations and an opportunity to go back to menu or to continue the game.

****