Оглавление

[**3** ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 2](#_Toc416725893)

[**3.1** Классы, реализующие блок пользовательского интерфейса 3](#_Toc416725894)

[3.1.1 Класс ConversationController 3](#_Toc416725895)

[3.1.2 Класс MusicController 5](#_Toc416725896)

[3.1.3 Класс UploadController 6](#_Toc416725897)

[3.1.4 Класс UsersController 7](#_Toc416725898)

[**3.2** Классы, реализующие блок аутентификации 8](#_Toc416725899)

[3.2.1 Класс SongPictureGetter 9](#_Toc416725900)

[3.2.2 Класс UploadService 9](#_Toc416725901)

[3.2.3 Класс StatisticsService 11](#_Toc416725902)

[3.2.4 Класс WallService 12](#_Toc416725903)

[3.2.5 Класс LastFmService 12](#_Toc416725904)

[3.2.6 Класс VkSongInfoGetter 14](#_Toc416725905)

[**3.3** Классы, реализующие блок работы с файловой системой на стороне клиент 14](#_Toc416725906)

[3.3.1 Класс AccountController 14](#_Toc416725907)

[3.3.2 Класс MailSender 15](#_Toc416725908)

[**3.4** Классы, реализующие объектную модель базы данных 16](#_Toc416725909)

[3.4.1 Класс ActivateCode 16](#_Toc416725910)

[3.4.2 Класс Comment 17](#_Toc416725911)

[3.4.3 Класс DigitalFile 18](#_Toc416725912)

[3.4.4 Класс FileVsTag 19](#_Toc416725913)

[3.4.5 Класс Folder 19](#_Toc416725914)

[3.4.6 Класс FolderVsFile 20](#_Toc416725915)

[3.4.7 Класс FolderVsUser 20](#_Toc416725916)

[3.4.8 Класс FriendLink 21](#_Toc416725917)

[3.4.9 Класс Message 21](#_Toc416725917)

[3.4.10 Класс Tag 21](#_Toc416725917)

[3.4.11 Класс User 21](#_Toc416725917)

[**3.5** Классы, реализующие блок доступа к базе данных 22](#_Toc416725918)

[3.5.1 Класс DdmRepository 22](#_Toc416725919)

[**3.6** Классы, реализующие блок веб-сервиса 22](#_Toc416725918)

[**3.7** Классы, реализующие блок работы с файловой системой на стороне сервера 22](#_Toc416725918)

[**3.8** Классы, реализующие блок работы с виртуальными каталогами 22](#_Toc416725918)

# **3** ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Рассмотрим подробно функционирование программы. Для этого проведем анализ всех классов, которые входят в состав кода программы, и рассмотрим назначение всех методов, свойств и переменных класса.

Условно в программе можно выделить две части:

* Серверная часть (написана с использованием wcf сервиса на объектно ориентированном языке C#);
* Клиентская часть (написана с использованием технологии Wpf на объектно ориентированном языке C#).

Структура серверной части состоит из трех основных уровней:

* класс, реализующий веб сервис, на который ложиться обработка запросов;
* классы, реализующие работу с файловой системой на стороне сервера;
* классы, реализующие работу базы данных (блок классов представляющих объектную модель базы данных и блок классов, обеспечивающий взаимодействие с таблицами базы дан);

Структура клиентской части состоит из нескольких уровней:

* классы, реализующие работу с файловой системой на стороне клиента. Они отвечают за работу с файлами в файловой системе(создание, удаление, расчет контрольной суммы);
* классы, реализующие пользовательский интерфейс. Эти классы наследуются от системного класса Page или Window, а также несколько классов-надстроек над элементами интерфейса, они нужны для удобного построения пользовательского интерфейса в приложении Wpf;
* классы, реализующие работу с виртуальными каталогами. Эти классы нужны для построения дерева физических каталогов на ряду с добавленными к ним виртуальными каталогами.
* классы, реализующие аутентификацию пользователя. Они нужны для регистрации новых пользователей, входа уже существующих, и активации существующих аккаунтов пользователей;
* вспомогательные классы. Это набор классов широкого спектра применения, они используются также и в серверной части. К ним относятся классы для подсчета контрольной суммы, серелизации и десериализации, записи и чтения из файла и т.д.;

## **3.1** Классы, реализующие блок пользовательского интерфейса

Классы, реализующие этот блок, предназначены для удобного отображения интерфейса приложения. Часть классов наследована от системных классов Page или Window, часть являются надстройками над элементами интерфейса. Почти все из них взаимодействуют с классами из других уровней клиентского приложения. По сути эти классы являются самым верхним уровнем клиентского приложения.

Классы этого блока довольно различны. Каждый из них является узко специализированным, и отвечает за определенную часть интерфейса

### 3.1.1 Класс ConversationController

Представляет собой класс, реализующий обработку запросов связанных с беседами. Ответами на запросы обычно являются либо объекты в формате Json либо html разметка. В методах данного класса отсутствует бизнес логика, он служит только посредником информации между клиентом и сервером. Является наследником класса Controller.

Поля:

* \_conversationService *–* поле типаIConversatioService, содержит сервис которому передается управление после того как класс контроллера получил запрос.

Методы:

* ConversationController (IConversationService conversationService) – конструктор класса;
* ActionResult ViewConversations() *–* метод, возвращающий html разметку с беседами;
* ActionResult ViewEditConversations()–метод, возвращающий html разметку с редактированеим беседам;
* ActionResult ViewConversation()– метод, возвращающий html разметку с содержащую разметку просмотра беседы;
* ActionResult ChangeAvatarModal() – метод, возвращающий html разметку с содержащую диалоговое окно для смены аватарки бесед;
* JsonResult GetMyConversations(int type) – метод, возвращающий беседы текущего авторизованного пользователя в формате Json , входной параметр тип возвращаемой беседы;
* JsonResult GetMusicConversations() – метод, возвращающий беседы все открытые музыкальные беседы.
* JsonResult GetMusicConversations() – метод, возвращающий беседы все открытые музыкальные беседы;
* ActionResult AddConversation (ConversationViewModel conversationViewModel) – метод, добавляющий новую беседу принимает объект беседы и передает его сервису \_conversationService;
* JsonResult AddOrGetDialog(string userId) - метод, добавляющий новый диалог текущего пользователя с пользователем, идентификатор которого пришел во входном параметре;
* ActionResult RemoveConversation(ConversationViewModel conversationViewModel)- метод, принимающий беседу которую необходимо удалить;
* JsonResult GetConversationPeople(ConversationViewModel conversationViewModel)- метод, принимающий в качестве параметра беседу и возвращающий сообщения данной беседы;
* JsonResult GetConversationMessages(ConversationViewModel conversationViewModel)- метод, принимающий в качестве параметра беседу и возвращающий сообщения данной беседы;
* ActionResult AddMessageToConversation(string text, string conversationId, List<Song> songs) - метод, добавляющий сообщение в беседу. Принимает текст сообщения идентификатор беседы и список песен ассоциированных с сообщением;
* ActionResult RemoveMessageFromConversation(string messageId, string conversationId) – метод, удаляющий сообщение из беседы. Принимает идентификатор беседы и идентификатор сообщения, подлежащего удалению;
* ActionResult AddUserToConversation(string userId, string conversationId) – метод, добавляющий пользователя в беседу. Принимает идентификатор беседы и идентификатор пользователя, подлежащего добавлению;
* ActionResult RemoveUserFromConversation(UserViewModel user, string conversationId) – метод, удаляющий пользователя из беседы. Принимает пользователя, подлежащего удалению и идентификатор беседы;
* JsonResult GetMyNotReadMessagesCount() – метод, возвращающий число непрочитанных сообщений во всех беседах в которых состоит пользователь ;
* JsonResult ReadConversationMessages(string conversationId) – метод, служащий для прочтения всех сообщений в беседы идентификатор которой передан;
* JsonResult ReadConversationMessages(string conversationId) – метод, служащий для прочтения всех сообщений в беседы идентификатор которой передан;
* JsonResult UpdateConversationCurrentSong(string conversationId, string songId)– метод, служащий для обновления текущей песни беседы, идентификаторы беседы и песни являются принимаемыми параметрами;
* JsonResult GetConversation(string conversationId) – метод, возвращающий беседу по текущему

### 3.1.2 Класс MusicController

Представляет собой класс, реализующий обработку запросов связанных с песнями. Ответами на запросы обычно являются либо объекты в формате Json либо html разметка. В методах данного класса отсутствует бизнес логика, он служит только посредником информации между клиентом и сервером. Является наследником класса Controller.

Поля:

* \_musicService – поле типа IMusicService, содержащее сервис с бизнес логикой;

Методы:

* MusicController(IMusicService musicService) – конструктор класса;
* ActionResult Player() *–* метод, возвращающий html разметку содержащую страницу плеера;
* ActionResult ViewSong() – метод, возвращающий html разметку содержащую страницу песни;
* ActionResult ViewSongsModal() *–*метод, возвращающий html разметку содержащую диалоговое окно с песнями пользователя;
* JsonResult GetSongs() *–* метод, возвращающий cписок песен в формате Json;
* JsonResult GetUserSongs(string userId) *–* метод, возвращающий список песен в формате Json принадлежащих пользователю с принимаемым идентификатором;
* JsonResult GetSong(string songId) – метод, возвращающий песню, ассоциированную с принимаемым идентификатором;
* JsonResult GetMySongs() – метод, возвращающий список песен текущего авторизованного пользователя;
* JsonResult RemoveSong(SongViewModel song) – метод, удаляющий песню из списка из аудиозаписей пользователя, принимает запись для удаления;
* JsonResult AddToMyMusic(SongViewModel song) – метод, добавляющий песню в список аудизозаписей пользователя, принимает запись для добавления;
* JsonResult ListenedSong(string songId) – метод, добавляющий песню связанную с принимаемым идентификатором в список прослушанных песен пользователя.

### 3.1.3 Класс UploadController

Представляет собой класс реализующий обработку запросов связанных с загрузкой контента на сайт. Ответами на запросы обычно являются либо объекты в формате Json либо html разметка. Является наследником класса Controller.

Поля:

* \_uploadService – поле типа IUploadService, содержащее сервис с бизнес логикой по загрузке данных;
* \_playlistService– поле типа IUploadService, содержащее сервис с бизнес логикой по редактированию плейлистов;

Методы:

* ActionResult Upload() – метод, возвращающий html разметку содержащую страницу загрузки музыки;
* JsonResult UploadImage(HttpPostedFileBase file) – метод, производящий загрузку картинки с аватаркой пользователя на сервер принимает файл типа HttpPostedFileBase;
* JsonResult UploadConversationImage(HttpPostedFileBase file, string conversationId) – метод, производящий загрузку картинки с аватаркой беседына сервер принимает файл типа HttpPostedFileBase и идентификатор беседы для которой загружается аватарка;
* ActionResult UploadSong(HttpPostedFileBase file) *–* метод, производящий загрузку песни на сервер принимает файл типа HttpPostedFileBase являющийся аудиозаписью в формате mp3;
* JsonResult GetSongsVk (VkUserModel model) – метод, производящий загрузку песни на сервер с сервисов социальной сети «вконтакте »принимает объект класса VkUserModel содержащего данные учетной записи «вконтакте» возвращает список песен привязанных к этой учетной записи;
* JsonResult SaveSongs(List<SongViewModel> songs) – метод, сохраняющий список песен загруженных из сети «вконтакте» песни добавляются в специальный плейлист;
* JsonResult SaveSong(SongViewModel song) – метод, сохраняющий песню песен загруженную из сети «вконтакте» песня добавляются в специальный плейлист.

### 3.1.4 Класс UsersController

Представляет собой класс, реализующий обработку запросов связанных с пользователями. Ответами на запросы обычно являются либо объекты в формате Json либо html разметка. В методах данного класса отсутствует бизнес логика, он служит только посредником информации между клиентом и сервером. Является наследником класса Controller.

Поля:

* \_userService– переменная класса IUserService, хранящая сервис для работы с пользователями.

Методы:

* UsersController (IUserService userService) – конструктор класса;
* JsonResult GetUsers()– метод, возвращающий html разметку со страницей пользователей;
* JsonResult Index()– метод, возвращающий html разметку домашней страницы пользователя;
* JsonResult GetUser(string id) – метод, возвращающий html разметку с диалоговым окном пользователей;
* JsonResult GetMyInfo() – метод, возвращающий html разметку со страницей пользователя;
* JsonResult UpdateUser(UserViewModel userInfo) – метод, обновляющий данные о текущем пользователе;
* JsonResult SearchUsers(UserSearchModel searchModel) – метод производящий поиск по пользователям входным параметром является объект типа UserSearchModel в нем содержатся свойства по которым будет осуществлен поиск возвращается список объектов в формате Json;
* JsonResult UpdateUserVkInfo(VkUserModel userInfo) – метод, принимающий информацию о учетной записи «вконтакте»;
* ActionResult ViewFriends() – метод, возвращающий html разметку со страницей друзей;
* JsonResult UpdateUserCurrentSong(string songId) – метод, принимающий идентификатор песни которую сейчас слушает авторизованный пользователь;
* ActionResult AddFriend(UserViewModel friend) – метод принимающий пользователя которого необходимо добавить в друзья текущего авторизованного пользователя;
* ActionResult RemoveFriend(UserViewModel friend) – метод принимающий пользователя которого необходимо удалить из друзья текущего авторизованного пользователя;;
* ActionResult GetUserFriends(string id) – метод, возвращающий список друзей пользователя заданного входным параметром список представляет собой Json объект;

## **3.2** Классы реализующие блоки загрузки, работы с внешними сервисами, обработки статистики

### 

### 3.2.1 Класс SongPictureGetter

Представляет собой статический класс, реализующий функционал получения картинки обложки альбома из различных источников, таких как веб-сервис, теги внутри mp3 файла

Методы:

* static AlbumTrackInfo GetPictureByWebService(Tag tag, string titleVk, string artistVk, string fileName) – метод, получающий путь к обложке альбома используя веб сервис last.fm,поиск осуществляется по всем входными параметрам которыми являются объект класса тег из которого выделяются поля исполнителя ,названия песни , и название альбома также принимает название файла и название исполнителя и песни полученные от сервиса «вконтакте». Выходным параметром является объект класса AlbumTrackInfo который содержит путь к картинке;
* static void GetAndSavePictureByTag(Tag tag, string songCoverPath, string songId) – метод, достающий обложку альбома из тега. Обложка предсавляет собой массив байт который после преобразования выдается в виде файла который сохранаятеся на диске.
* static AlbumTrackInfo CheckContent(string artist, string album, string track) – метод делающий запрос на сервер сети «last.fm» используя входные параметры название альбома, трэка и исполнителя.

### 3.2.2 Класс UploadService

Представляет собой класс, реализующий сохранение песен, обложек альбомов, аватарок пользователей управление ему передается из класса UploadController.

Поля:

* \_musicRepository – поле типа IMusicRepository, реализация этого интерфейса реализует логику работу с базой данных в частности с таблицей песен (Songs);
* \_playlistRepository – поле типа IPlaylistRepository поле, реализация этого интерфейса реализует логику работу с базой данных в частности с таблицей плейлистов (Playlists);
* \_userRepository – поле типа IUserRepository поле, реализация этого интерфейса реализует логику работу с базой данных в частности с таблицей пользователей Users;
* \_conversationRepository – поле типа IConversationRepository поле, реализация этого интерфейса реализует логику работу с базой данных в частности с таблицами песен;

Методы:

* UploadService(IMusicRepository musicRepository,IPlaylistRepository playlistRepository,IUserRepository userRepository, IConversationRepository conversationRepository)– конструктор класса;
* void UploadSong(string fileExtension, string fileName, string pathSong, string songId, string absoluteSongCoverPath, string userId) – метод, cохраняющий загруженную песню и ее обложку альбома в файловую систему по пути pathSong, а также сохраняющий путь к этой песне в базу данных песню в базу данных переменным начальные значения;
* void SaveSongFromVkUpdatePicture(SongViewModel song, string userid)– метод, предназначенный для сохранения получения обложки песни загруженной из сети «вконтакте».
* void UploadUserImage(string imagePath, string userId) – метод сохраняющий в базу данных относительный путь к картинке пользователя идентификатор которого задается параметром;
* void UploadConversationImage(string imagePath, string conversationId) – метод, сохраняющий в базу данных относительный путь к картинке беседы идентификатор которой задается вторым параметром;
* SongInfo GetLyricsAndSongInfoByVK(File audioFile, string fileName) – метод, получающий информацию о песне вместе с ее текстом. Информация принимается от сервиса «вконтакте » входными данными является файл с тегами а также имя файла;
* string SongAlbumPicturePathToDb(File audioFile, string saveSongCoverPath, string songId, string songAlbumPicturePathToDb, string titleVk, string artistVk, string filename, ref string content) – метод, сбрасывающий текущую позицию указателя в текстовом поле;

### 3.2.3 Класс StatisticsService

Представляет собой класс, содержащий логику по обработке статистики прослушиваемых и добавленных классов. Содержит в себе интерфейсы классов блока работы с базой данных в частности классы. Реализует интерфейс IStatisticsService.

Поля:

* \_statisticsRepository *–* поле типа IStatisticsRepository, реализация данного интерфейса предоставляет методы обеспечивающие выборки из таблиц базы данных Songs и ListenedSongs;
* \_musicRepository – поле типа IMusicRepository, реализация данного интерфейса предоставляет методы обеспечивающие выборки из таблиц базы данных Songs;

Методы:

* StatisticsService(IStatisticsRepository statisticsRepository,IMusicRepository musicRepository) – конструктор класса;
* IEnumerable<SongViewModel>GetFavouriteSongs(string userId) – метод, возвращающий коллекцию песен чаще всего прослушиваемых пользователем. Пользователь задается вторым входным параметром*;*
* IEnumerable<SongViewModel> GetFavouriteSongs() –метод, возвращающий коллекцию песен чаще всего прослушиваемых всеми пользоваетелями сети;
* IEnumerable<SongViewModel>GetLastListenedSongs(string userId) – метод, возвращающий коллекцию песен которые были недавно просушены пользователем заданным вторым параметром;
* IEnumerable<SongViewModel> GetLastAdded() – метод, возвращающий коллекцию песен которые были недавно добаввлены в сеть.
* IEnumerable<SongChartModel> GetChartData(string songId) – метод, возвращающий коллекцию данных для построения графика прослушивания песни. Идентификатор песни является входным параметром. В данном случае возвращается коллекция объектов типа SongChartModel содержащих дату прослушивания песни и количество прослушиваний.

### 3.2.4 Класс WallService

Представляет собой класс, содержащий методы работы со стенкой пользователя.

Поля:

* \_wallRepository – поле типа IWallRepository, содержащее методы работы с таблицами базы данных(WallItems, WallItemsSongs);
* \_userRepository *–* поле типа IUserRepository, содержащее методы работы с таблицами базы данных.

Методы:

* WallService( IWallRepository wallRepository, IUserRepository userRepository) – конструктор класса;
* List<WallItemViewModel> GetWall(string userId) – метод возвращающий коллекцию элементов стенки пользователя заданного входным параметром;
* WallItemViewModel GetWallItem(string userId, int wallItemId) – метод возвращающий элемент стенки пользователя заданного входным параметром.Первый параметр является идентификатором пользоваетеля, второй параметр является идентфикатором возвращаемого элемента;
* void AddWallItem(WallItemViewModel wallItemView) – метод реализующий функционал по доабвлению новой записи на стенку поьзователя объект для доалвения является входным параметр;
* void RemoveWallItem(string userId, int wallItemId) – метод реализующий функционал по удалению новой записи с стенки пользователя идентификатор пользователя с чьей стеник буде производится удалнеия и идентфификатор записи подлежащей удалению;

### 3.2.5 Класс LastFmService

Представляет собой класс, взаимодействующий с api сервиса «last.fm». Служит для получения обложек альбомов по названиям песни альбома или исполнителя.

Поля:

* const \_ApiKey – поле , содержащее строковую константу ключ api для работы с сервисом;
* const \_ mainUrl– поле, содержащее строковую константу путь к веб сервису;
* const methodTrackGetInfo – поле, содержащее строковую константу парметра для составления строки запроса;
* const methodAlbumGetInfo – поле, содержащее строковую константу парметра для составления строки запроса;
* const methodTrackSearch – поле, содержащее строковую константу парметра для составления строки запроса;
* const methodTrackGetTags – поле, содержащее строковую константу парметра для составления строки запроса;
* const methodArtistGetTags – поле, содержащее строковую константу парметра для составления строки запроса;

Методы:

* GetSessionKey() – метод получает ключ сессии для последующей работы с веб сервисом;
* List<string> GetTag(string artist) – метод возвращающий коллекцию тегов ассоциированных с исполнителем заданным входным параметром;
* List<string> GetTag(string artist, string track) – – метод возвращающий коллекцию тегов ассоциированных с исполнителем и названием трека заданными входными параметрами;
* List<string> GetTagsFromXml (string url, List<string> result) – метод, достающий названия тегов из xml файла получаемого по заданному url;
* AlbumTrackInfo GetAlbumInfo(string artist, string album) – метод, получающий путь к обложке альбома по названию исполнителя и навзанию альбома;
* AlbumTrackInfo GetTrackInfo(string artist, string track) – метод, получающий путь к обложке альбома по названию исполнителя и названию трека;
* AlbumTrackInfo SearchTrackInfo(string track) – метод, получающий информацию о треке по его названию. Метод получает название трека, делает запрос к сервису сети «last.fm», получает ответ в виде xml, разбирает его по узлам и достает путь к обложке альбома и информацию о песне.
* string CalculateMD5Hash(string input) – метод, вычисляющий Md5 хэш, входной строки служащей параметром для запроса на получение ключа сессии к сервису last.fm.После получения ключа сессии возможна работа всех остальных методов приведенных выше;

### 3.2.6 Класс VkSongInfoGetter

Представляет собой класс, взаимодействующий с api сервиса «вконтакте». Служит для информации о тексте песни названии песни, названии исполнителя.

Поля:

* \_vkApi– поле, типа VkApi является объектно-ориентированной оберткой над методами предоставляемыми сервисом, методы данного класса будут использоваться в методах класса VkSongInfoGetter;

Методы:

* VkSongInfoGetter(VkApi vkApi) – конструктор объекта;
* VkSongInfoGetter(string login, string password) – конструктор объекта;
* string GetSongLyrics(long? lyricsId) – – метод возвращающий текст песни по ее идентификатору;
* SongInfo GetSongInfo(string songTitle) – метод, возвращающий экземпляр объекта SongInfo найденный по названию песни;
* List<SongInfo> GetSongs(bool findLyrics) – метод, возвращающий коллекцию песен привязанных к учетной записи пользователя (логин и пароль, передаваемые в конструктор класса). Булевский флаг указывает на то, будет ли проводится поиск текстов песен или нет;
* void GetLyricsFromAllAudios(IEnumerable<Audio> audios, SongInfo info) – метод, получающий текста песен ко всем объектам в принимаемой коллекции;

## **3.3** Классы реализующие блок авторизации

### 3.3.1 Класс AccountController

Представляет собой класс, отвечающий за регистрацию новых пользователей в системе, также за выполнение входа в систему существующих пользователей. Содержит методы регистрации, входа в систему. Напрямую работает с базой данных. Наследуется от класса Controller.

Поля:

* \_userRepository – поле типа IUserRepository, использующееся для обеспечения доступа к таблицам пользователей в базе данных;

Методы:

* AccountController(IUserRepository serRepository)– конструктор класса;
* ActionResult Login() – метод, возвращающий html разметку для страницы входа в систему;
* ActionResult Register() – метод, возвращающий html разметку для страницы регистрации в систему;
* Task<JsonResult> Login(LoginViewModel model) – метод, осуществляющий вход в систему, принимает LoginViewModel в экземпляре данного класса содержится логин и пароль, если эта модель проходит валидацию и такой пользователь найден в базе данных возвращается флаг Success, что подтверждает вход на сайт;
* ActionResult CheckLogin() – метод, осуществляющий проверку авторизован ли пользователь, зашедший на сайт, если текущая сессия пользователя еще не завершена то вернется флаг LoggedIn принимающий положительное значение, если данный флаг не вернется, то пользователь будет направлен на страницу Login где ему будет предложено выполнить вход снова;
* Task<JsonResult> Register(RegisterViewModel model) – метод, осуществляющий регистрацию новых пользователей в системе, принимает экземпляр класса RegisterViewModel, полями которого являются логин, пароль и почтовый адрес, если пользователя с таким логином еще не было в базе данных создаётся не активированная учетная запись и посылается письмо на почту указанную при регистрации;
* Task<JsonResult> ConfirmEmail(string token, string email) – метод, служащий, для активации пользователей проходящий по ссылке в письме, в виде токена принимается уникальный идентификатор пользователя, также принимает почтовый адрес, который сравнивается с адресом закрепленными за этой учетной запись и если они совпадают учетная запись пользователя, раз блокируется для последующей работы.

### 3.3.2 Класс MailSender

Представляет собой вспомогательный класс для класса AccountController, реализующий только один метод по отправке письма на почту.

Методы:

* static void SendEmailMessage(ApplicationUser user, string url) – статический метод по отправке письма на почту введенную пользователем при регистрации. Метод формирует адрес отправителя, и адрес получателя а также тело сообщения после чего осуществляет отправку по Smtp протоколу.

## **3.4** Классы, реализующие объектную модель базы данных

Данные классы необходимы при использовании Entity Framework для работы с базой данных в частности одного из трех подходов Code First когда разработчик сначала описывает классы – модели, а фреймворк по этим классам строит базу данных. Все класс содержат автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

### 3.4.1 Класс ActivateCode

Класс модель таблицы ActivateCode, все поля и свойства класса описывают колонки данной таблицы.

Свойства:

* Id – обертка над полем типа Guid, хранящее первичный ключ для данной таблицы;
* Code – обертка над полем типа String, хранящее код для активации аккаунта. Код генерируется используя четыре символа из id, что гарантирует приемливую уникальность;
* Date – обертка над полем типа DateTime, хранящим дату создания кода активации. Используется для отслеживания актуальности кода на данный момент;
* UserId – обертка над полем типа Guid, хранящее id пользователя, для активации которого должен быть использован данный код;

### 3.4.2 Класс Comment

Класс модель таблицы Comment, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в таблице Comment;
* Message – обертка над полем типа string хранящее текст коментария к файлу;
* UserID – обертка над полем типа Guid, хранящим id пользователя, который оставил коментарий;
* FileId – обертка над полем типа Guid, хранящим id файла к которому оставлен коментарием;
* Index – обертка над полем типа int, хранящим индекс сообщения, для корректного вывода истории коментариев;

### 3.4.3 Класс DigitalFile

Класс модель таблицы DigitalFile, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в таблице DigitalFile;
* Name – обертка над строковым полем, хранящим имя файла. Имя храниться также с его расширением;
* CheckSum – обертка над полем типа String, хранящим чек-сумму для данного файла. Чек-сумма нужна для отслеживания версионности файла;

### 3.4.4 Класс FileVsTag

Класс модель таблицы FileVsTag, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в таблице FileVsTag;
* FileId – обертка над полем типа Guid, хранящим id файла, к которому прикреплен тэг;

### TagId – обертка над полем типа Guid, хранящим id тэга, прикрепленного к файлу;

### 3.4.5 Класс Folder

Класс модель таблицы Folder, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в таблице Folder;
* Name – обертка над полем типа String, хранящим имя папки;
* ParrentId – обертка над полем типа Guid, хранящим id паррент-каталога, в который вложен данный каталог. Если это поле пусто, данный каталог является корневым;
* CreateUserId – обертка над полем типа Guid, хранящим id пользователя, который создал данную папку;

### 3.4.6 Класс FolderVsFile

Класс модель таблицы FolderVsFile, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в таблице FolderVsFile;
* FolderId – обертка над полем типа Guid, хранящим id папки, к которому закреплен файл;
* FileId – обертка над полем типа Guid, хранящим id файла, который закреплен за каталогом;

### 3.4.7 Класс FolderVsUser

Класс модель таблицы FolderVsUser, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в таблице FolderVsUser;
* FolderId – обертка над полем типа Guid, хранящим id папки, которая закреплена за пользователем;
* UserId – обертка над полем типа Guid, хранящим id пользователя, за которым закреплена папка;

### 3.4.8 Класс FriendLink

Класс модель таблицы FriendLink, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в таблице FriendLink;
* UserId – обертка над полем типа Guid, хранящим id пользователя, к которому в друзья добавлен друг;
* FriendId – обертка над полем типа Guid, хранящим id пользователя, который добавлен в друзья первому пользователю;

### 3.4.9 Класс Message

Класс модель таблицы Message, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в таблице Message;
* Text – обертка над полем типа String, хранящим текст сообщения, которое отсылается пользователю;
* UserID – обертка над полем типа Guid, хранящим id пользователя, который отправляет сообщение;
* DestUserId – обертка над полем типа Guid, хранящим id пользователя, которому отсылается сообщение;
* FileId – обертка над полем типа Guid, хранящим id файла, который вложен в сообщение;

### 3.4.10 Класс Tag

Класс модель таблицы Tag, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в таблице Tag;
* Name – обертка над полем типа String, хранящим имя тега в текстовом виде;
* UserId – обертка над полем типа Guid, хранящим id пользователя, который создал этот тэг;

### 3.4.11 Класс User

Класс модель таблицы User, все поля и свойства класса описывают колонки данной таблицы. Класс содержит автоматические свойства C#, при объявлении которых компилятор создает закрытое анонимное резервное поле, которое доступно только с помощью методов get и set свойства. При описании свойств будет приведено описание полей, оборачиваемых данными свойствами.

Свойства:

* Id – обертка над полем типа Guid, хранящим уникальный идентификатор, данное поле будет являться первичным ключом в данной таблице User;
* Login – обертка над полем типа String, хранящее имя, используя которое пользователь будет входить в систему;
* Password – обертка над полем типа String, хранящим пароль данного аккаунта. Пароль будет использован для авторизации пользователя в системе;
* Email – обертка над полем типа String, хранящим адрес электронной почты пользователя. Почта будет использована для активации аккаунта, а также для рассылки различной служебной информации;
* IsActivate – обертка над полем типа Bool, хранящим статус данного аккаунта;

## **3.5** Классы, реализующие блок доступа к базе данных

В данных классах реализован непосредственный доступ к базе данных. Вызов методов данного класса происходит в веб-сервисе. Методы реализуют функционал по удалению, добавлению, модификации строк из базы данных, также производят выборки данных из всех таблиц базы данных. Основная задача данного класса обеспечить промежуточный слой между базой данных и другими блоками приложения.

### 3.5.1 Класс DdmRepository

Класс в обеспечивающий доступ ко всем таблицам базы данных. Методы класса производят выборки данных, их модификацию, добавление новых данных в эти таблицы. Реализует интерфейс IDdmRepository.

Методы:

* ActivateCode CreateActivateCode(Guid userId) – создает новой код активации, заполняет его поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию объекта;
* Comment CreateComment(string message, Guid userID, Guid fileId, int index) – создает новое сообщение, заполняет его поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию объекта;
* DigitalFile CreateDigitalFile(string name) – создает новый объект, для описания файла, заполняет его поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию объекта;
* FileVsTag CreateFileVsTag(Guid fileID, Guid tagId) – создает новую ассоциацию между файлам и тэгом, заполняет её поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию объекта;
* Folder CreateFolder(string name, Guid parrentId, Guid createUserId) – создает новую папку, заполняет её поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию обьекта;
* FolderVsFile CreateFolderVsFile(Guid folderId, Guid fileId) – создает новую ассоциацию между папкой и файлом, заполняет её поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию обьекта;
* FolderVsUser CreateFolderVsUser(Guid folderId, Guid userId) – создает новую ассоциацию между папкой и пользователем, заполняет её поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию обьекта;
* FriendLink CreateFriendLink(Guid userId, Guid friendId) – создает новую ассоциацию между пользователями, заполняет её поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию обьекта;
* Message CreateMessage(string text, Guid userId, Guid descUserID, Guid fileId) – создает новое сообщение, заполняет его поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию обьекта;
* Tag CreateTag(string name, Guid userId) – создает новой тэг, заполняет его поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию обьекта;
* User CreateUser(string userName, string password, string email) – создает нового пользователя, заполняет его поля в соответствии со входными параметрами. Далее сохраняет созданный объект в базу данных. Возвращает копию обьекта;
* void UpdateFile(Stream fileStream, Guid fileId) – метод, который принимает стрим файла, и обновляет им уже существующий в базе файл. Для поиска файла используется его id;
* Stream GetFileStream(Guid fileId) – метод, который находит в базе данных информацию по файлу, используя его id, далее считывает его в стрим, и возвращает его;
* ActivateCode GetActivateCode(Guid activateCode Id) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* Comment GetComment(Guid commentId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* DigitalFile GetDigitalFile(Guid digitalFileId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* FileVsTag GetFileVsTag(Guid fileVsTagId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* Folder GetFolder(Guid folderId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* FolderVsFile GetFolderVsFile(Guid folVsFileId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* FolderVsUser GetFolderVsUser(Guid folVsUserId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* FriendLink GetFriendLink(Guid friendLinkId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* Message GetMessage(Guid messageId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* Tag GetTag(Guid TagId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* User GetUser(Guid userId) – используя id выбирает из базы данных объект, далее создает его копию и возвращает ее в вызывающий метод;
* List<ActivateCode> GetAllActivateCode() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<Comment> GetAllComment() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<DigitalFile> GetAllDigitalFile() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<FileVsTag> GetAllFileVsTag() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<Folder> GetAllFolder() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<FolderVsFile> GetAllFolderVsFile() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<FolderVsUser> GetAllFolderVsUser() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<FriendLink> GetAllFriendLink() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<Message> GetAllMessage() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<Tag> GetAllTag() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* List<User> GetAllUser() – создает пустой список объектов, далее в цикле проходит по всем элементам таблицы, создает их локальную копию и добавляет ее в список. Возвращает список в вызывающюю функцию;
* void ActivateUser(Guid userId) –выбирает пользователя из таблицы используя его идентификатор, затем выставляет полу isActivate в true. После этого данный пользователь может войти в систему;
* List<Tag> GetAllTag(Guid fileId) – возвращает все тэги, связанные с данным файлом;