Компьютерная академия «ШАГ»

Днепропетровский филиал

Кафедра Разработки программного обеспечения

КУРСОВОЙ ПРОЕКТ

«Zireael Messenger Web»

Студентов группы: ЕКО-16-П-1  
Евсеенко Н.В.

Самарского В.О.

Научный руководитель:  
Волынец Евгений Анатольевич

Днепр 2019 г.

# Аннотация

Пояснительная записка содержит описание алгоритма программы «Zireael Messenger Web» и сервиса для работы с приложением. Чтобы облегчить работу с программой, в содержание введена структура системы и руководство пользователя.

Пояснительная записка оформлена в соответствии с предъявляемыми требованиями. Записка выполнена на 30 листах формата А4. Содержание пояснительной записки соответствует заданию, а разделы соответствуют содержанию.

Оглавление

[Аннотация 2](#_Toc533401261)

[Оглавление 3](#_Toc533401262)

[Введение 4](#_Toc533401263)

[Требования к функциональным характеристикам клиент-приложения 5](#_Toc533401264)

[Требования к техническим характеристикам 6](#_Toc533401265)

[Требования к программным характеристикам 7](#_Toc533401266)

[Проектирование системы 8](#_Toc533401267)

[Проектирование интерфейса 8](#_Toc533401268)

[Проектирование классов для клиент-приложения 10](#_Toc533401269)

[Проектирование базы данных 13](#_Toc533401270)

[Проектирование сервиса 17](#_Toc533401271)

[Функции сервиса 18](#_Toc533401272)

[Руководство пользователя 23](#_Toc533401273)

[Дополнения 30](#_Toc533401274)

# Введение

Темой данного курсового проекта является разработка простого приложения для обмена сообщениями и сервиса для обработки информации и взаимодействия с БД.

Чем доступней для людей становятся компьютерные технологии, тем больше появляется непрофессиональных пользователей. Поэтому основной задачей каждого программиста является создание интуитивно понятного интерфейса программы. Это задача так же будет выполнена в этом курсовом проекте.

Целью курсового проекта является:

* научиться создавать качественное и надежное программное обеспечение для обработки большого количества данных
* разработать техническую документацию к разработанному программному обеспечению
* Получить и закрепить навыки в работе с такими технологиями как ASP.NET MVC, WCF, Entity Framework, ADO.NET
* Получить и закрепить навыки в работе с пакетом программ Microsoft Visual Studio 2017 и Microsoft SQL Server

Результатом курсового проекта должна быть полноценная база данных для хранения данных, решающая поставленную задачу, приложение для работы с этой базой данных, а также техническая документация к указанному программному обеспечению.

# Требования к функциональным характеристикам клиент-приложения

В процессе курсового проектирования необходимо разработать ПО для обмена сообщениями. Данное программное обеспечение должно иметь простой и удобный интерфейс.

Так же необходим сервис для обработки информации и взаимодействия с базой данных.

Программное обеспечение должно выполнять следующие функции:

* Регистрация новых пользователей
* Вход в аккаунт
* Восстановление пароля
* Отправка текстовых сообщений
* Чтение сообщений
* Создание групп до 1000 пользователей
* Добавление контактов
* Настройка программы и профиля

Сервис должен выполнять следующие функции:

* Обработка запросов клиентов
* Работа с базой данных
* Уведомление пользователям об изменении профиля(на почту)

# Требования к техническим характеристикам

Для работы программного обеспечения необходимо, что бы используемые вами технические средства соответствовали ниже перечисленным характеристикам:

* Компьютер типа IBM PC или ноутбук
* Процессор Pentium Celeron (Pentium IV и выше), AMD Athlon (и выше), с тактовой частотой не меньше 1,6 ГГц
* 1024 МБ ОЗ
* HHD или SSD со свободным местом не меньше 100 МБ
* Устройства ввода и манипуляций
* Видеоадаптер с 512 МБ памяти и более и поддержкой Microsoft DirectX 9 с драйвером WDDM
* Персональный компьютер должен быть подключен к глобальной сети интернет

# Требования к программным характеристикам

Для эксплуатации данного программного обеспечения необходимо, что бы на компьютере были установлены:

* операционная версия Windows 8+
* .NET Framework 4.7+
* Для работы компьютер должен быть подключен к глобальной сети Интернет

Для работы сервиса необходимо:

* операционная версия Windows 8+
* NET Framework 4.7+
* MS SQL Server
* Сервер должен быть подключен к глобальной сети Интернет

# Проектирование системы

## Проектирование интерфейса

Основной задачей каждого программиста является создание интуитивно понятного интерфейса программы, потому что удобный интерфейс всегда сделает работу пользователя приятней.

Для создания сайта была выбрана платформа ASP.NET MVC.

Платформа ASP.NET MVC представляет собой фреймворк для создания сайтов и веб-приложений с помощью реализации паттерна MVC.

Концепция паттерна (шаблона) MVC (model - view - controller) предполагает разделение приложения на три компонента:

**Контроллер** (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.

**Представление** (view) - это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.

**Модель** (model) представляет класс, описывающий логику используемых данных.

Общую схему взаимодействия этих компонентов можно представить следующим образом:



В этой схеме модель является независимым компонентом - любые изменения контроллера или представления не затрагивают модель. Контроллер и представление являются относительно независимыми компонентами, и нередко их можно изменять независимо друг от друга.

Благодаря этому реализуется концепция **разделение ответственности**, в связи с чем легче построить работу над отдельными компонентами. Кроме того, вследствие этого приложение обладает лучшей тестируемостью. И если нам, допустим, важна визуальная часть или фронтэнд, то мы можем тестировать представление независимо от контроллера. Либо мы можем сосредоточиться на бэкэнде и тестировать контроллер.

Конкретные реализации и определения данного паттерна могут отличаться, но в силу своей гибкости и простоты он стал очень популярным в последнее время, особенно в сфере веб-разработки.

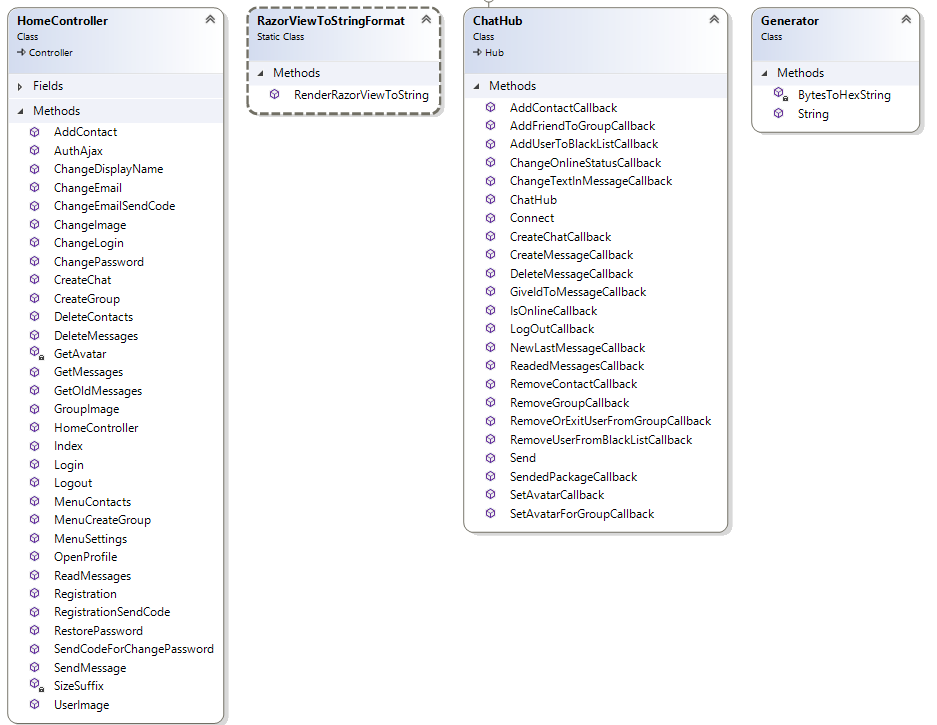
Свою реализацию паттерна представляет платформа ASP.NET MVC. 2013 год ознаменовался выходом новой версии ASP.NET MVC - MVC 5, а также релизом Visual Studio 2013, которая предоставляет инструментарий для работы с MVC5.

Хотя во многих аспектах MVC 5 не слишком сильно будет отличаться от MVC 4, многое из одной версии вполне применимо к другой, но в то же время есть и существенные отличия:

* В MVC 5 изменилась концепция аутентификации и авторизации. Вместо SimpleMembershipProvider была внедрена система ASP.NET Identity, которая использует компоненты OWIN и Katana.
* Для создания адаптивного и расширяемого интерфейса в MVC 5 используется css-фреймворк Bootstrap
* Добавлены фильтры аутентификации, а также появилась функциональность переопределения фильтров
* В MVC 5 также добавлены атрибуты маршрутизации

Это наиболее важные нововведения в MVC 5. Кроме того, есть еще ряд менее значимых, например, использование по умолчанию Entity Framework 6, некоторые изменения при создании проекта (концепция One ASP.NET), дополнительные компоненты и т.д.

## Проектирование классов для ASP.NET MVC



public static class RazorViewToStringFormat

string RenderRazorViewToString(Controller controller, string viewName, object model)

Конвертирует View в html код.

class ChatHub : Hub, ICeadChatServiceCallback

Реализует коллбек функции

public class Generator

string String(int size)

Генерирует случайную строку указанной длины

public class HomeController : Controller

Task<JsonResult> DeleteMessages(string messagesIdJson)

[POST]Обрабатывает запрос на удаление сообщения, возвращает уведомление об успешном удалении сообщений.

Task<JsonResult> GetMessages(int? groupId)

[POST]Возвращает последние n сообщений из группы с id groupId (n хранится в конфиге сервера)

Task<JsonResult> GetOldMessages(int? groupId, int? lastMessageId)

[POST]Возвращает n сообщений которые были отправлены перед lastMessageId в группе с id groupId (n хранится в конфиге сервера)

Task ReadMessages(int? groupId)

[POST]Уведомляет сервер что пользователь прочитал все сообщения в группе с id groupId

Task<JsonResult> SendMessage(string text, int groupId, long hash)

[POST]Оправляет сообщение от текущего пользователя в группу с id groupId и текстом text

Task<JsonResult> Logout()

[POST]Уведомляет сервер что пользователь вышел из аккаунта

Task<ActionResult> UserImage(int userId = 0, string hash = "", int number = 0)

[GET]Получает аватарку пользователя #number с id userId, если не установлена, генерирует аватарку с первой буквой имени в круге

Task<ActionResult> GroupImage(int groupId = 0, string hash = "", int number = 0)

[GET]Получает аватарку группы #number с id groupId, если не установлена, генерирует аватарку с первой буквой названия в круге

Task<AvatarWCF> GetAvatar(bool isUserImage, int id, string hash, int avatarNumber = 0)

Вспомогательная функция которая вызывается в UserImage и GroupImage

Task<ActionResult> Index()

[GET]Возвращает страницу авторизации если пользователь не залогинен или основной интерфейс если залогинен.

ActionResult AuthAjax(string type = "log")

[POST]Возвращает одно из view(Логин, Регистрация, Забыл пароль), в зависимости от того что пришло в type

Task<JsonResult> Login(string login, string password)

[POST]Выполняет вход в аккаунт с данными login и password

string SizeSuffix(long value)

Функция преобразовывает биты в читаемый вид(байты/килобайты/мегабайты…)

Task<JsonResult> MenuContacts()

[POST]Возвращает view с контантами

Task<JsonResult> MenuCreateGroup()

[POST]Возвращает view с создание группы

Task<JsonResult> MenuSettings()

[POST]Возвращает view с настройками

Task<JsonResult> OpenProfile(string login)

[POST]Возвращает view с профилем пользователя с логином login

Task<JsonResult> AddContact(string login)

[POST]Добавляет пользователя с логином login в контакты

Task<JsonResult> DeleteContacts(string idArr)

[POST]Удаляет контакты из json списка idArr

Task<JsonResult> CreateChat(int id)

[POST]Создаёт диалог с пользователем у которого id == id

Task<JsonResult> CreateGroup(string idArr, string groupName)

[POST]Создаёт группу с пользователями из json списка idArr, с названием groupName

Task<JsonResult> ChangeLogin(string login)

[POST]Изменяет логин текущего пользователя на login

Task<JsonResult> ChangeDisplayName(string displayName)

[POST]Изменяет имя текущего пользователя на displayName

Task<JsonResult> ChangePassword(string newPass, string repNewPass, string oldPass)

[POST]Изменяет пароль текущего пользователя на newPass

Task<JsonResult> ChangeEmailSendCode(string newEmail, string pass)

Отправляет на почту newEmailк случайный код для подтверждения

Task<JsonResult> ChangeEmail(string code)

[POST]Изменяет почту текущего пользователя на newEmail из функции ChangeEmailSendCode если правильно введён code

Task<JsonResult> ChangeImage()

[POST]Изменяет аватар текущего пользователя на переданный в запросе

Task<ActionResult> SendCodeForChangePassword(string loginOrEmail)

[POST]Отправляет на почту пользователя loginOrEmail код для восстановления пароля

Task<JsonResult> RestorePassword(string loginOrEmail, string pass, string repPass, string code)

[POST]Задаёт новый пароль pass пользователю loginOrEmail если правильно введён код с почты code

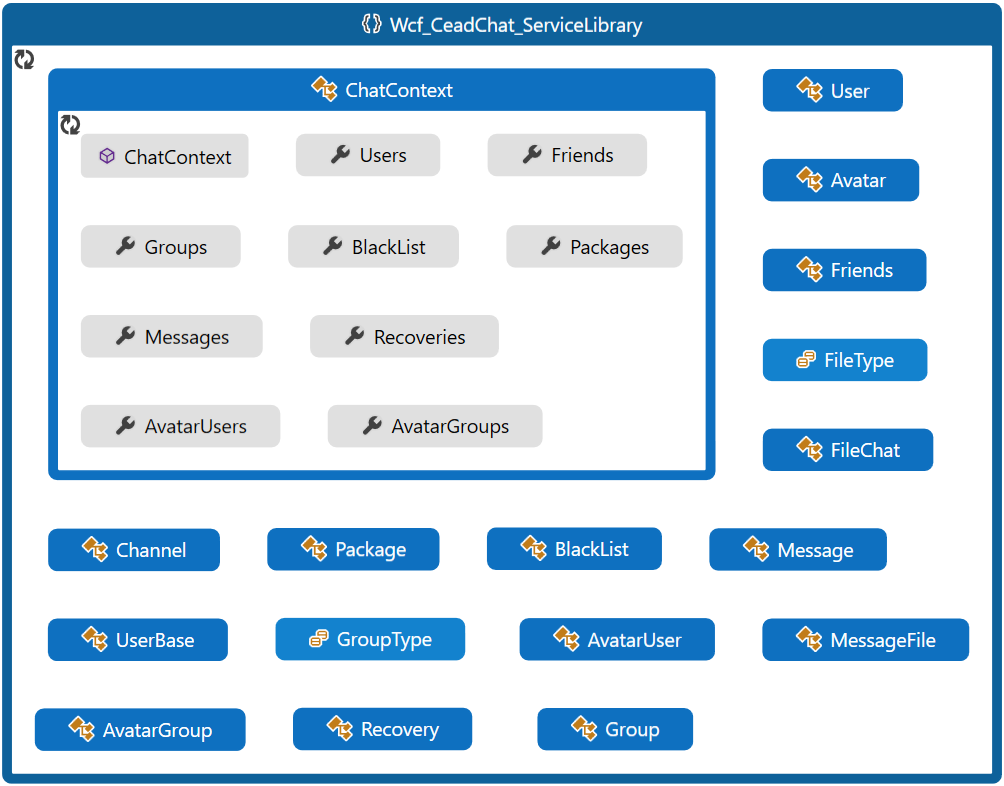
Task<JsonResult> RegistrationSendCode(string email)

[POST]Отправляет на почту код для подтверждения аккаунта при регистрации

Task<JsonResult> Registration(string login, string password, string repPassword, string email, string code)

[POST]Производит регистацию пользователя с логином login, паролем password, почтой email, если код подтверждения code введён правильно

## Проектирование базы данных



Класс **UserBase** представляет модель «пользователь» и имеет следующие свойства:

* public int Id – уникальный идентификатор пользователя
* public string Login – логин пользователя
* public string DisplayName – имя пользователя
* public DateTime LastTimeOnline – время последнего онлайна
* public bool IsOnline – онлайн

Класс **User** дополняет модель «пользователь» и имеет следующие свойства:

* public string Email – почтовый адрес пользователя
* public string PasswordHash – хеш пароля (Argon2)
* public ICollection<Group> Groups – чаты в которых состоит пользователь
* public DateTime DateCreated – дата создания аккаунта
* public string Token - токен авторизации
* public DateTime TokenDate – дата создания токена авторизации

Класс **Avatar** представляет модель «аватар» и имеет следующие свойства:

* public int Id – уникальный идентификатор картинки
* public byte[] SmallData – картинка 64х64
* public byte[] BigData – картинка 512х512
* public string Format – формат картинки
* public DateTime DateTime – дата загрузки картинки

Класс **AvatarUser** наследуется от Avatar представляет модель «аватар пользователя» и имеет следующие свойства:

* public UserBase User – владелец аватара

Класс **AvatarGroup** наследуется от Avatar представляет модель «аватар чата» и имеет следующие свойства:

* public Group Group – чат которому присвоен аватар

Класс **Friends** представляет модель «контакты» и имеет следующие свойства:

* public int Id – уникальный идентификатор пары
* public User Sender – пользователь у которого в контактах User2
* public User User2 – добавленный в контакты

Класс **Package** представляет модель «часть файла» и имеет следующие свойства:

* public int Id – уникальный идентификатор пакета
* public int Number – номер по порядку
* public byte[] Data - байты

Класс **FileChat** представляет модель «файл» и имеет следующие свойства:

* public int Id – уникальный идентификатор файла
* public string Name – название файла
* public int Lenght – длинна файла
* public string Hash – md5 хеш файла
* public string FullPath – полный путь к файлу
* public int CountPackages – кол-во частей
* public int CountReadyPackages – кол-во загруженных частей
* public List<Package> Packages – части файла

Перечисление **FileType** представляет собой модель «тип файла» содержит:

* Image – изображение
* File – файл другого типа

Перечисление **GroupType** представляет собой модель «тип чата» содержит:

* SingleUser - диалог
* MultyUser – многопользовательский чат
* Channel – канал

Класс **Group** представляет собой модель «чат» содержит свойства:

* public int Id – уникальный идентификатор чата
* public UserBase Creator – создатель чата
* public bool IsVisible -видимость чата
* public string Name – название чата
* public ICollection<User> Users – пользователи в чате
* public ICollection<Message> Messages – сообщения в чате
* public GroupType Type – тип чата
* public Message LastMessage – последнее сообщение

Класс **Message** представляет собой модель «сообщение» содержит свойства:

* public int Id – уникальный идентификатор чата
* public bool IsVisible – видимость сообщения
* public string Text – текст сообщения
* public DateTime DateTime – время отправки
* public bool IsRead – прочитано
* public bool IsChanged - изменено
* public Group Group – чат в котором отправлено сообщение
* public UserBase Sender - отправитель

Класс **MessageFile** наследуется от Message, представляет собой модель «файловое сообщение» содержит свойства:

* public FileChat File - файл
* public FileType Type – тип файла

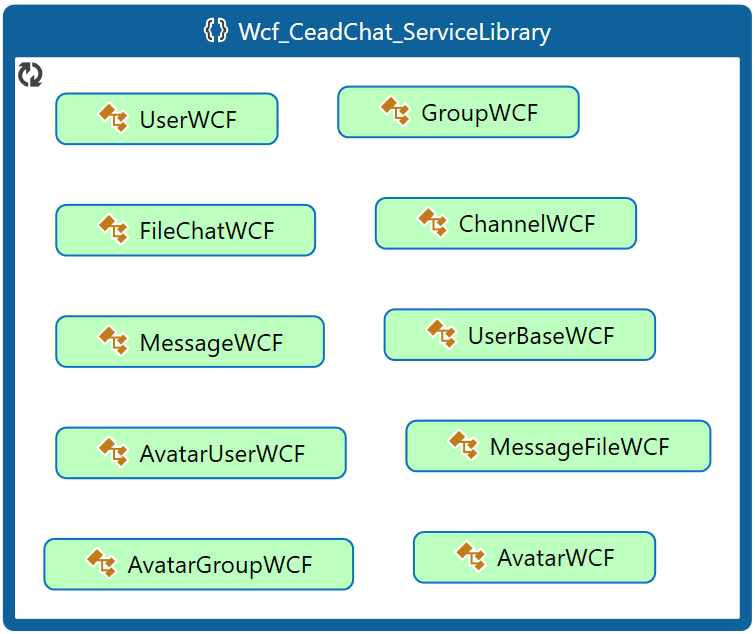
Класс **BlackList** представляет собой модель «черный список» содержит свойства:

* public int Id – уникальный идентификатор записи в чс
* public User Sender - блокирующий
* public User Blocked - заблокированный

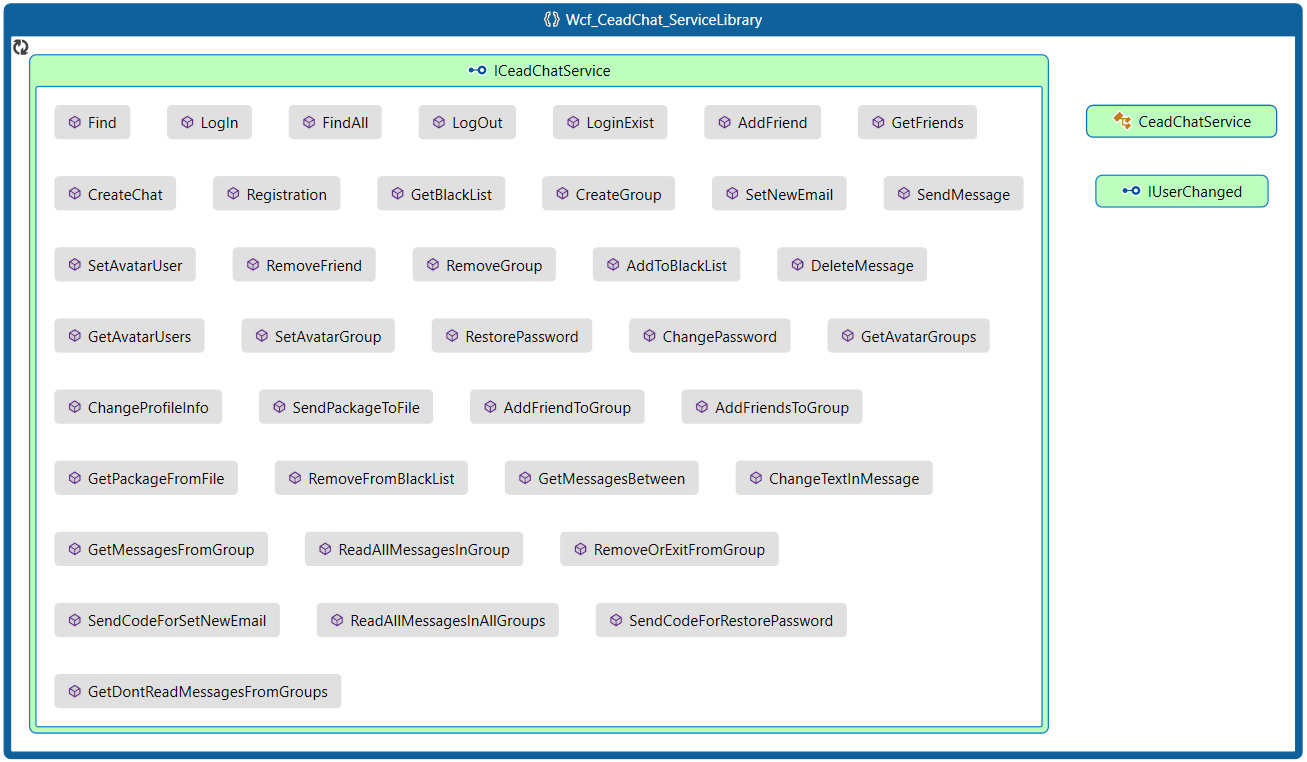
Класс **Recovery** представляет собой модель «код восстановления» содержит свойства:

* public int Id – уникальный идентификатор кода
* public DateTime RequestTime - время отправки кода
* public DateTime RecoveryTime - время приминения кода
* public string ConfirmationCode - код востановления
* public string NewInformation – новый почтовый адрес
* public bool IsWorking – работает
* public int UserId – пользователь который запросил код

## Проектирование сервиса

****

WCF не умеет передавать объекты с рекурсивными ссылками поэтому для транспортировки мы используем классы с постфиксом WCF которые дублируют классы базы данных, но не содержат рекурсивных ссылок.



### Функции сервиса

*public bool* ***AddFriend****(string userLogin)*

добавление пользователя c логином userLogin в список контактов

*public bool* ***AddFriendsToGroup****(GroupWCF group, IEnumerable<UserBaseWCF> friends)*

добавление контактов из списка friends в чат

*public bool* ***AddToBlackList****(UserBaseWCF user)*

блокировка пользователя user

*private void* ***CallUsersInGroup****(ICollection<User> usersCollection, Action<IUserChanged> action)*

отправка Callback всем пользователям из usersCollection

*public bool* ***ChangePassword****(string newPassword, string oldPassword)*

изменение пароля

*public bool* ***ChangeProfileInfo****(string displayName, string login)*

изменение имени и логина, если нужно изменить один параметр, второй можно передать null

*public bool* ***ChangeTextInMessage****(MessageWCF message)*

изменение текста сообщения *message*

*ChatContext* ***Context****(IUserChanged userChanged)*

получение данных из БД, если пользователь userChanged залогинен

*public bool* ***CreateChat****(UserBaseWCF addingUser)*

создание диалога с пользователем addingUser

*public bool* ***CreateGroup****(IEnumerable<UserBaseWCF> newUsers, string nameGroup)*

создание многопользовательского чата с именем nameGroup и пользователями newUsers

*private string* ***CreateToken****(UserBase user)*

генерация токена авторизации для пользователя user

*public bool* ***DeleteMessage****(MessageWCF message)*

удаление сообщения message

*public UserBaseWCF* ***Find****(string login)*

находит и возвращает пользователя по логину, если пользователя с таким логином нет, то вернется null

*public IEnumerable<UserBaseWCF>* ***FindAll****(string login, int maxCount)*

находит maxCount пользователей по Contains(login) и возвращает их

*private string* ***GeneratePasswordHash****(string password)*

генерирует хеш Argon2 для пароля password

*private List<UserBase>* ***GetAllFriends****(User sender)*

получение все связанных с sender пользователей(контакты и пользователи в чатах)

*public IEnumerable<AvatarGroupWCF>* ***GetAvatarGroups*** *(IEnumerable<GroupWCF> groups)*

получение аватаров групп из groups

*public IEnumerable<AvatarUserWCF>* ***GetAvatarUser s****(IEnumerable<UserBaseWCF> users)*

получение аватаров пользователей из users

*public IEnumerable<UserBaseWCF>* ***GetBlackList****()*

получение заблокированных пользователей

*public Dictionary<int, int>* ***GetDontReadMessagesFromGroups*** *(IEnumerable<int> groupsId)*

получение количиства непрочитанных сообщений для групп с id из groupsId

*public IEnumerable<UserBaseWCF>* ***GetFriends****()*

получение контактов для отправляющего

*public IEnumerable<MessageWCF>* ***GetMessagesBetween****(int groupId, int startIdx, int count)*

получает count сообщений из группы с id == groupId начиная со startIdx

*public IEnumerable<MessageWCF>* ***GetMessagesFromGroup****(GroupWCF group)*

получает все сообщения из группы group

*public Package* ***GetPackageFromFile****(int messageId, int packNumber)*

получение куска файла из сообщения с id == messageId и номером packNumber

*public UserWCF* ***LogIn****(string login, string password, string token)*

вход в аккаунт по логину и паролю либо по токену авторизации, если данные авторизации верны, то вернет пользователя иначе null

*public bool* ***LoginExist****(string login)*

проверка логина на существование, если логин существует вернется true

*public bool* ***LogOut****()*

пользователь вышел из аккаунта

*private void* ***NotificationAboutChangeOnlineStatus****(User user)*

изменился статус онлайн у пользователя user

*private string* ***RandomString****(int size)*

генерирует случайную строку размером size

*public bool* ***ReadAllMessagesInAllGroups****()*

прочитать все сообщения во всех группах для отправляющего

*public bool* ***ReadAllMessagesInGroup****(int groupId)*

прочитать все сообщения в группе

*public bool* ***Registration****(UserWCF newUser)*

зарегистрировать пользователя newUser

*public bool* ***RemoveFriend****(UserBaseWCF contact)*

удалить контакт contact

*public bool* ***RemoveFromBlackList****(UserBaseWCF user)*

разблокировать пользователя user

*public bool* ***RemoveGroup****(GroupWCF group)*

удаление чата group

*public bool* ***RemoveOrExitFromGroup****(int groupId, int userIdForRemove)*

удалить пользователя с id == userIdForRemove из группы с id == groupId

*public bool* ***RestorePassword****(string loginOrEmail, string code, string newPassword)*

установка пароля newPassword для пользователя с логином loginOrEmail, при условии, что он ввел правильный code присланный ему на почту

*public bool* ***SendCodeForRestorePassword****(string loginOrEmail)*

шлет на почту пользователя с логином loginOrEmail код для восстановления пароля

*public bool* ***SendCodeForSetNewEmail****(string newEmail, string password)*

если введен правильный пароль password, то на почту newEmail отправится код для подтверждения

*public int* ***SendMessage****(MessageWCF message, long hash)*

отправка сообщения message, hash нужен для верификации сообщения в callback

*void* ***SendMessageToMail****(string email, string title, string text)*

отправка сообщения text с заголовком title на почтовый адрес email

*public bool* ***SendPackageToFile****(int messageId, Package package)*

отправка куска файла Package из сообщения c id == messageId

*public bool* ***SetAvatarGroup****(AvatarGroupWCF avatar)*

установить аватар avatar группе avatar.Group

*public bool* ***SetAvatarUser****(AvatarUserWCF avatar)*

установить аватар пользователю avatar.User

*public bool* ***SetNewEmail****(string recoveryCode)*

если recoveryCode указан верно, пользователю, приславшему запрос устанавливается почтовый адрес закрепленный за этим кодом

*private void* ***Timer\_Elapsed****(object sender, ElapsedEventArgs e)*

проверка на месте ли онлайн пользователи, если callback не дошел, ставим пользователю офлайн статус

*private object* ***TryExecute****(ExceptionDelegate action, bool returnNull = false)*

action выполняется в обертке try, при ошибке вернет null или false

*private static void* ***WriteLog****(string message)*

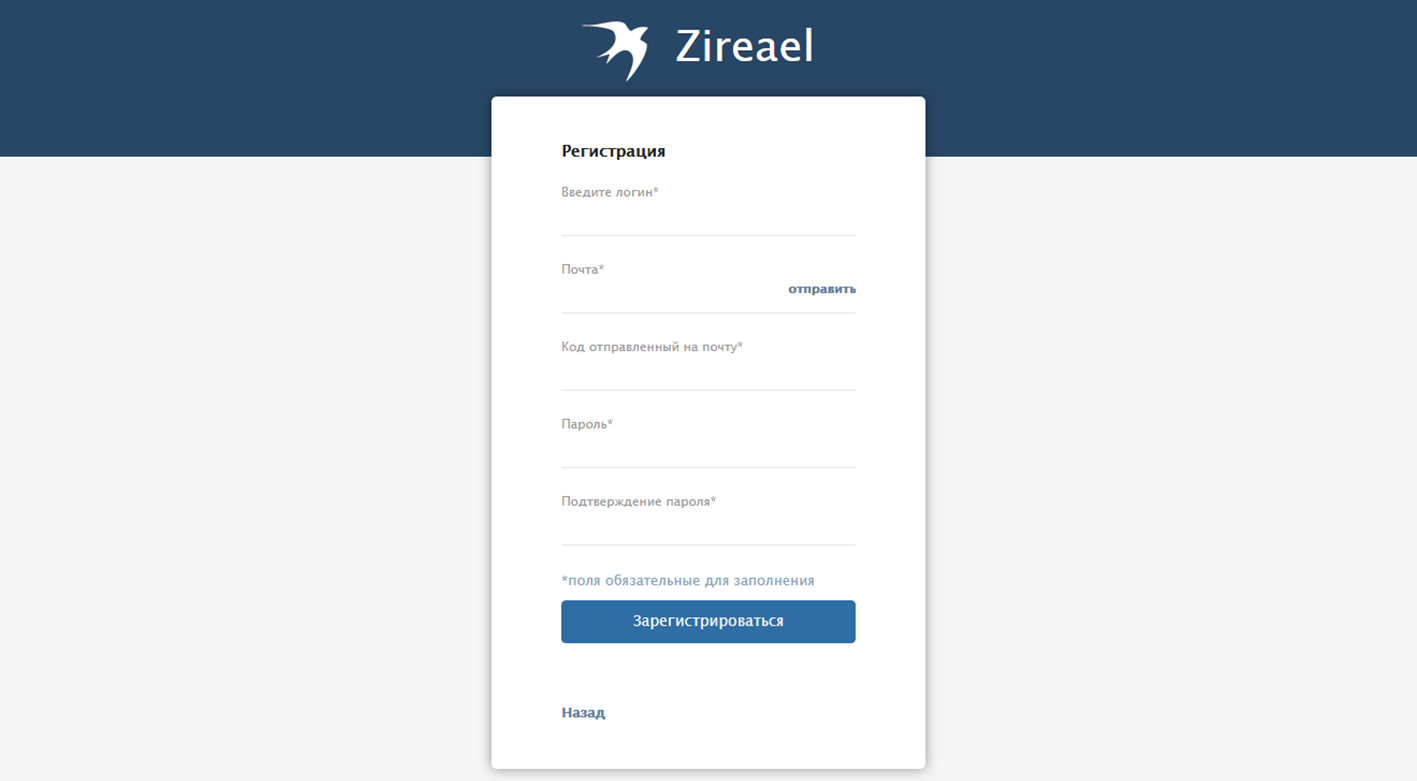
запись message в лог консоли

# Руководство пользователя

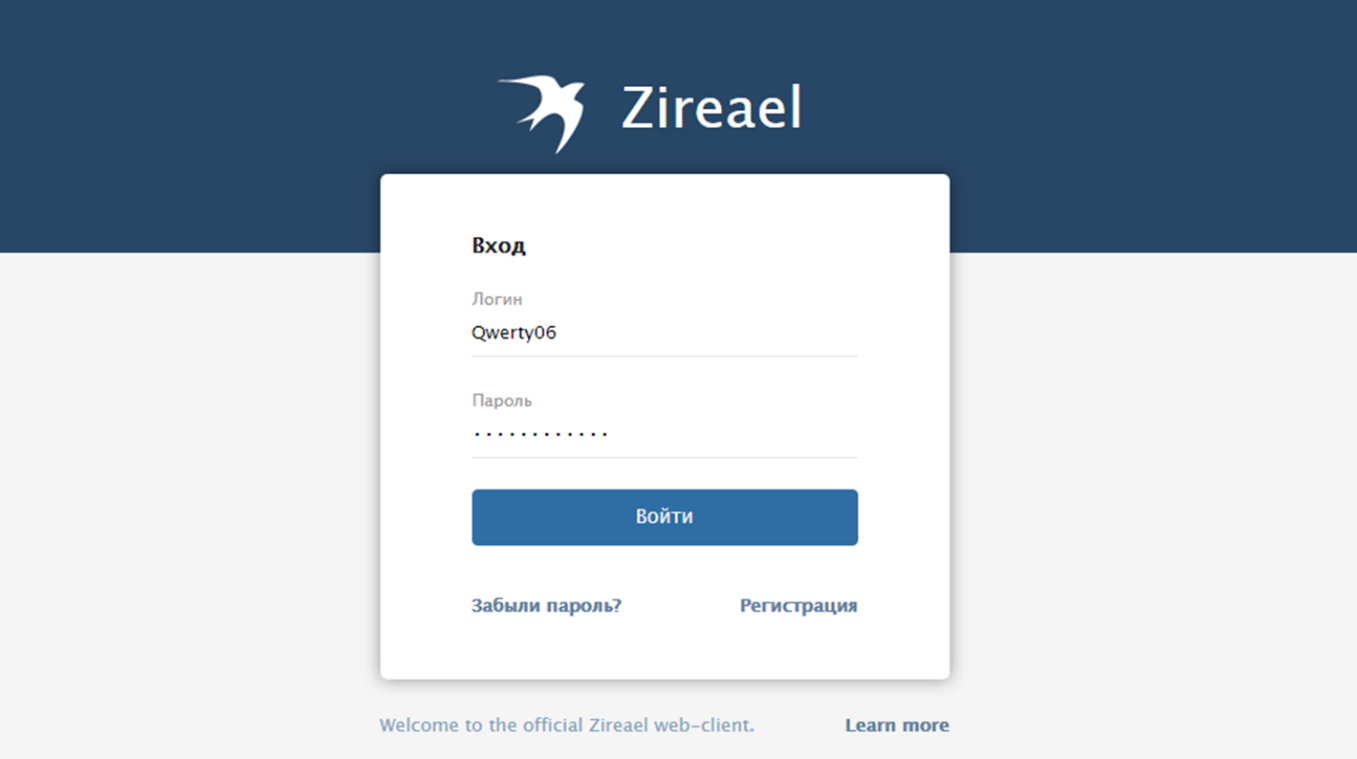
Запуск программы

Для запуска программы необходимо запустить файл с названием client.exe

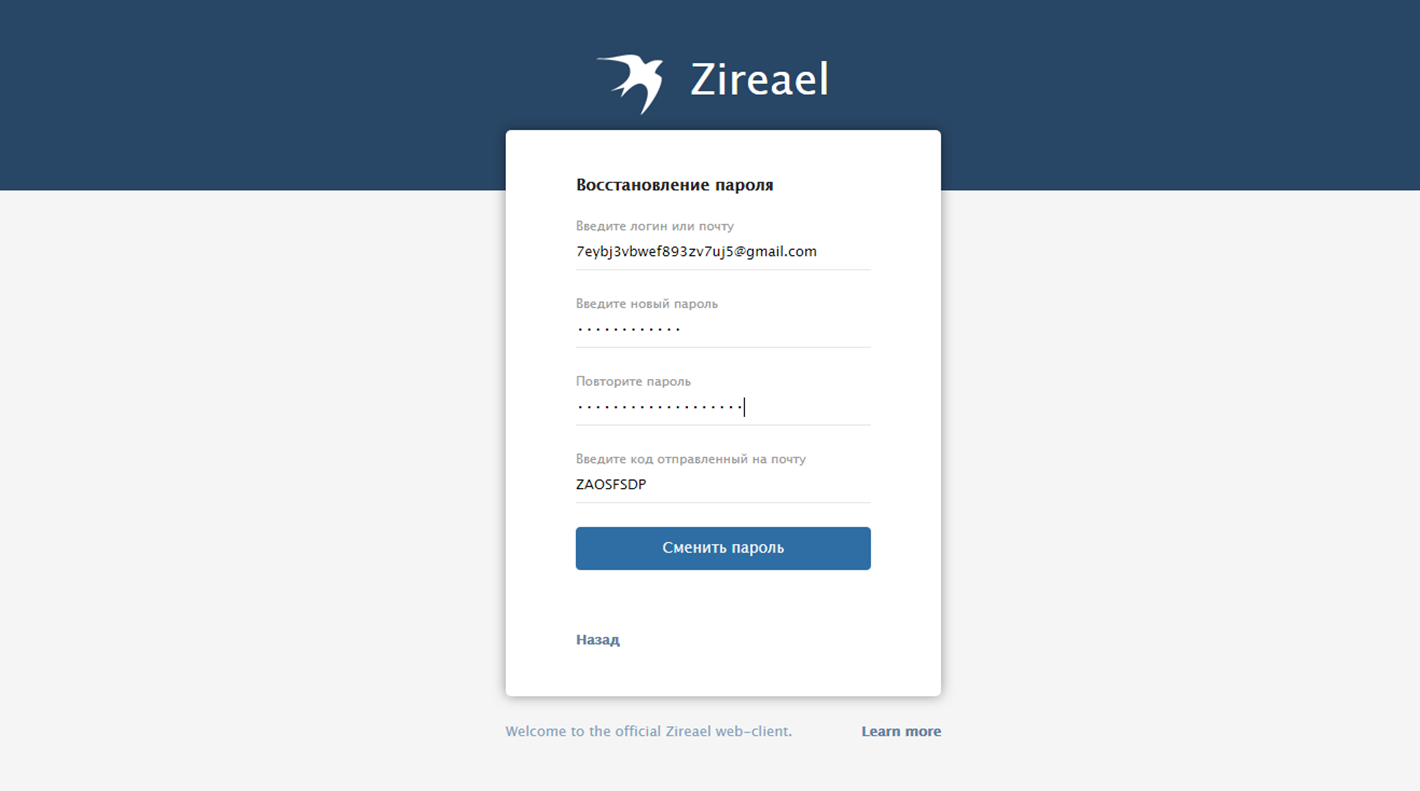
Регистрация



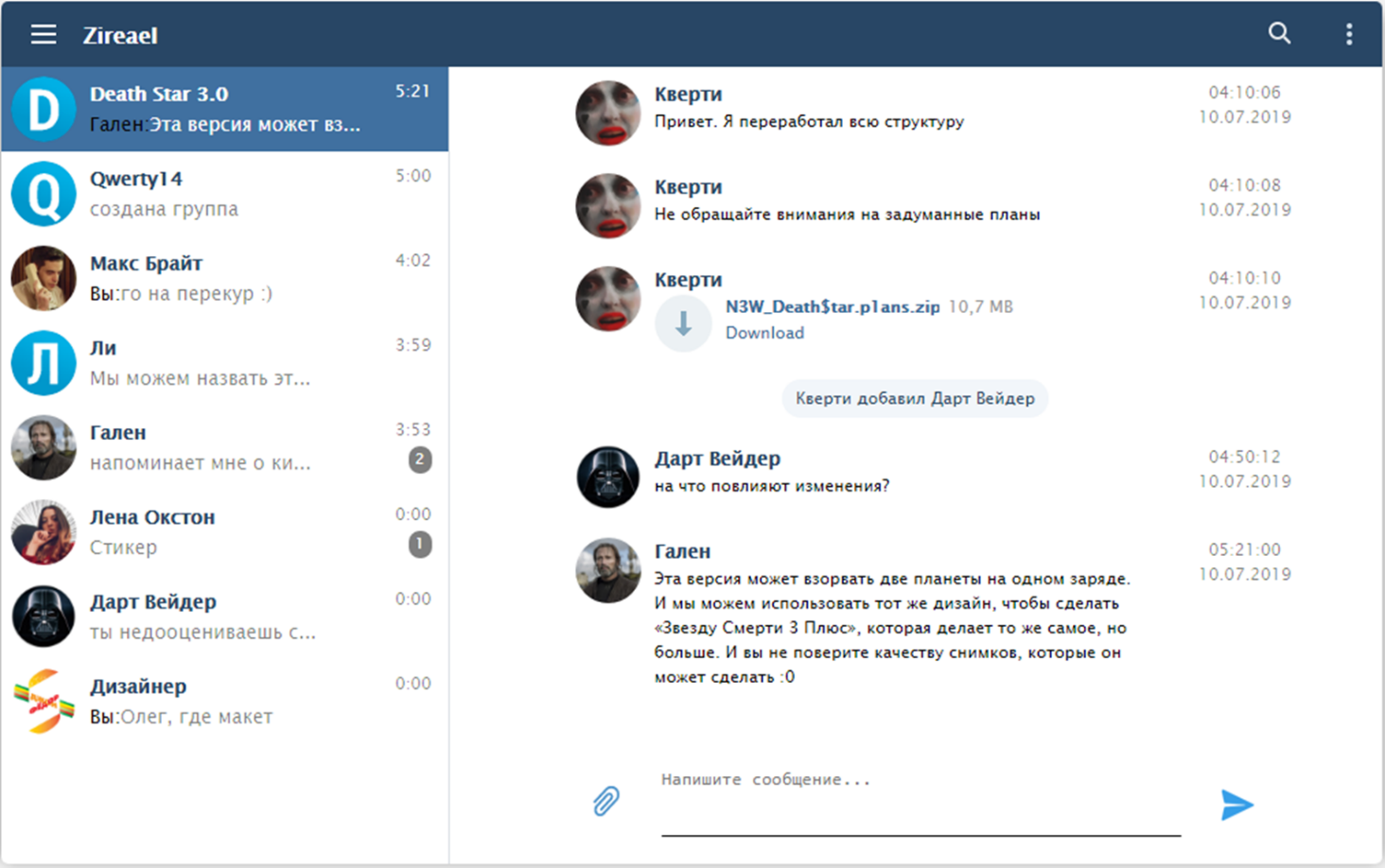
Вход



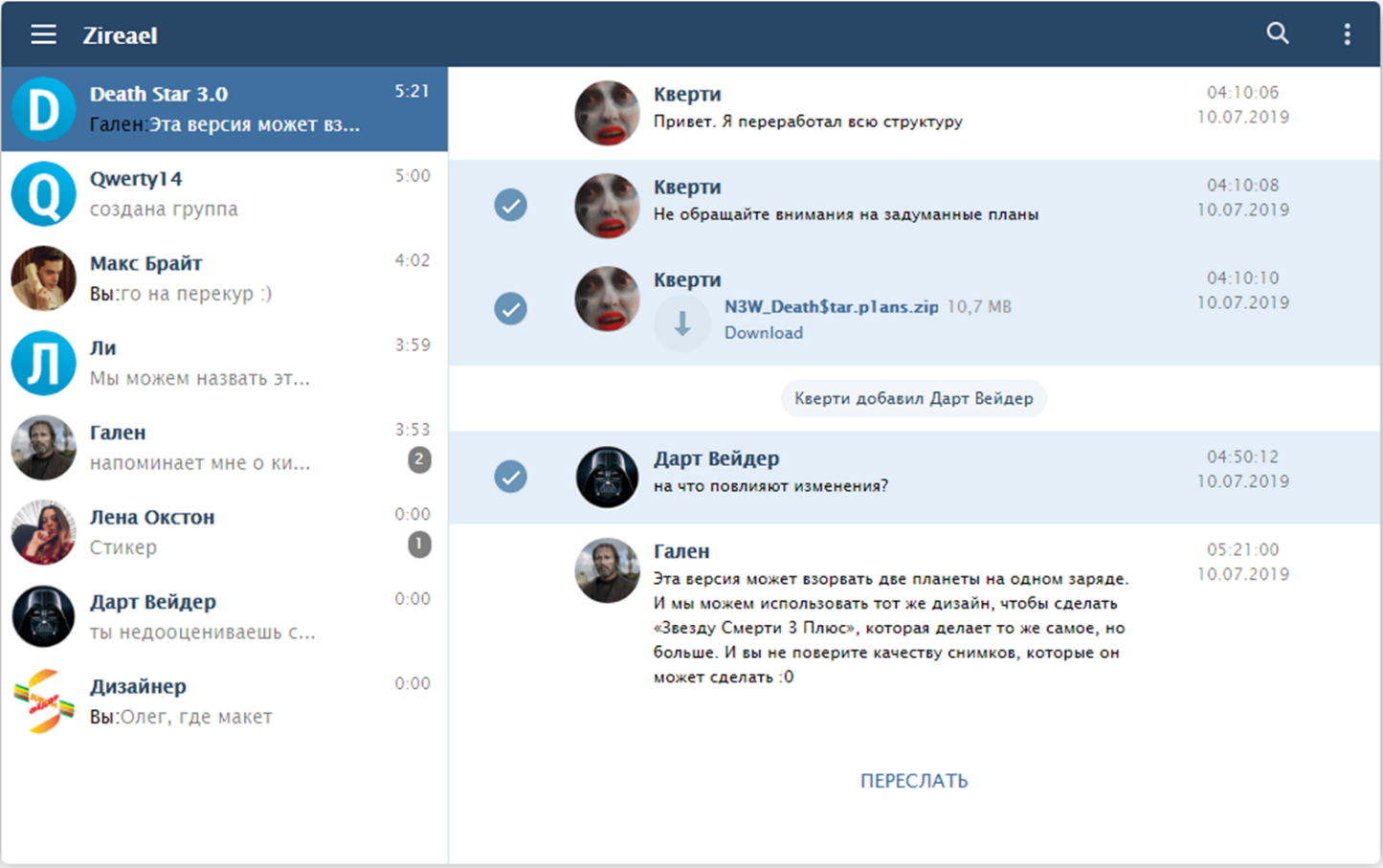
Восстановление пароля



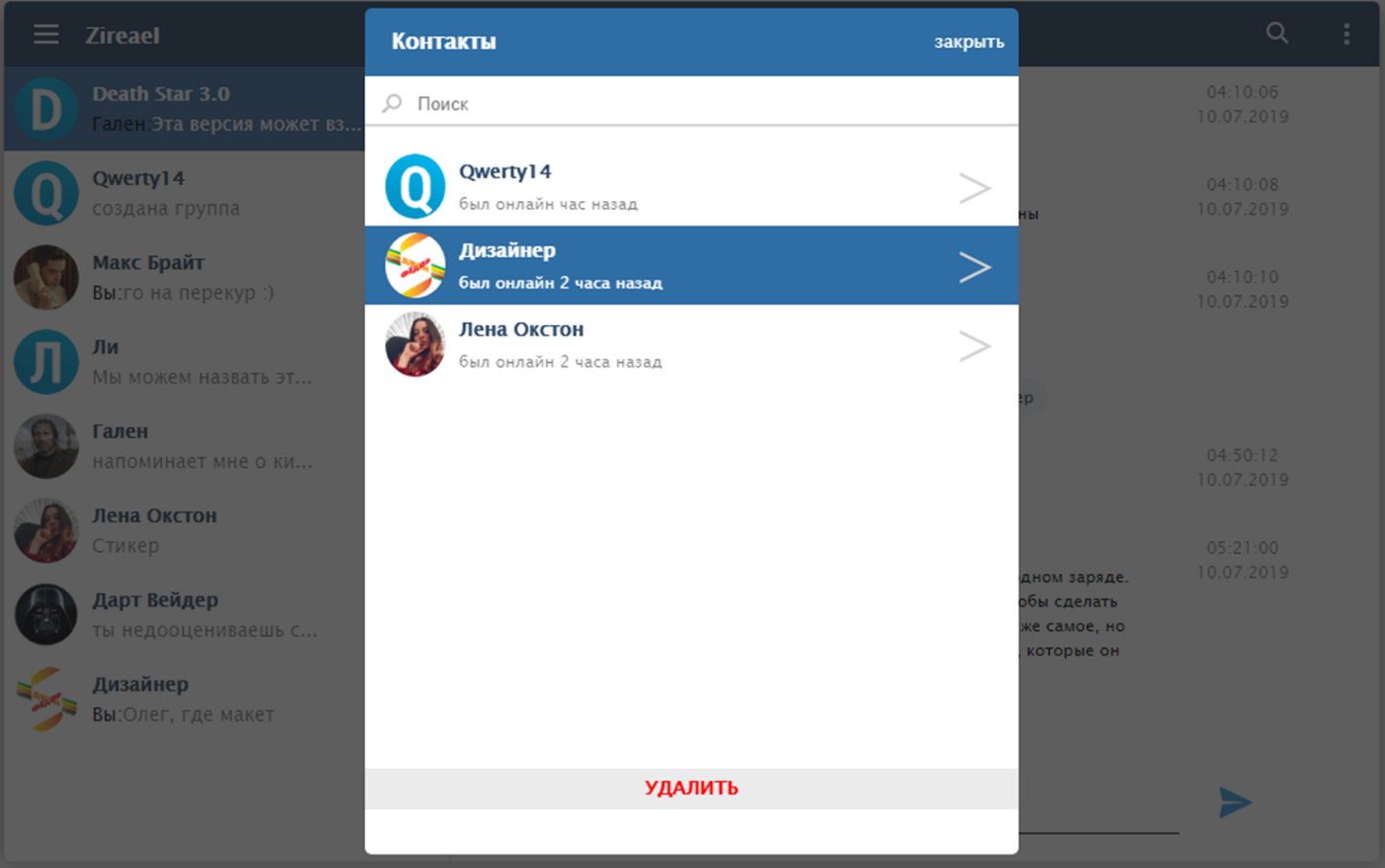
Интерфейс



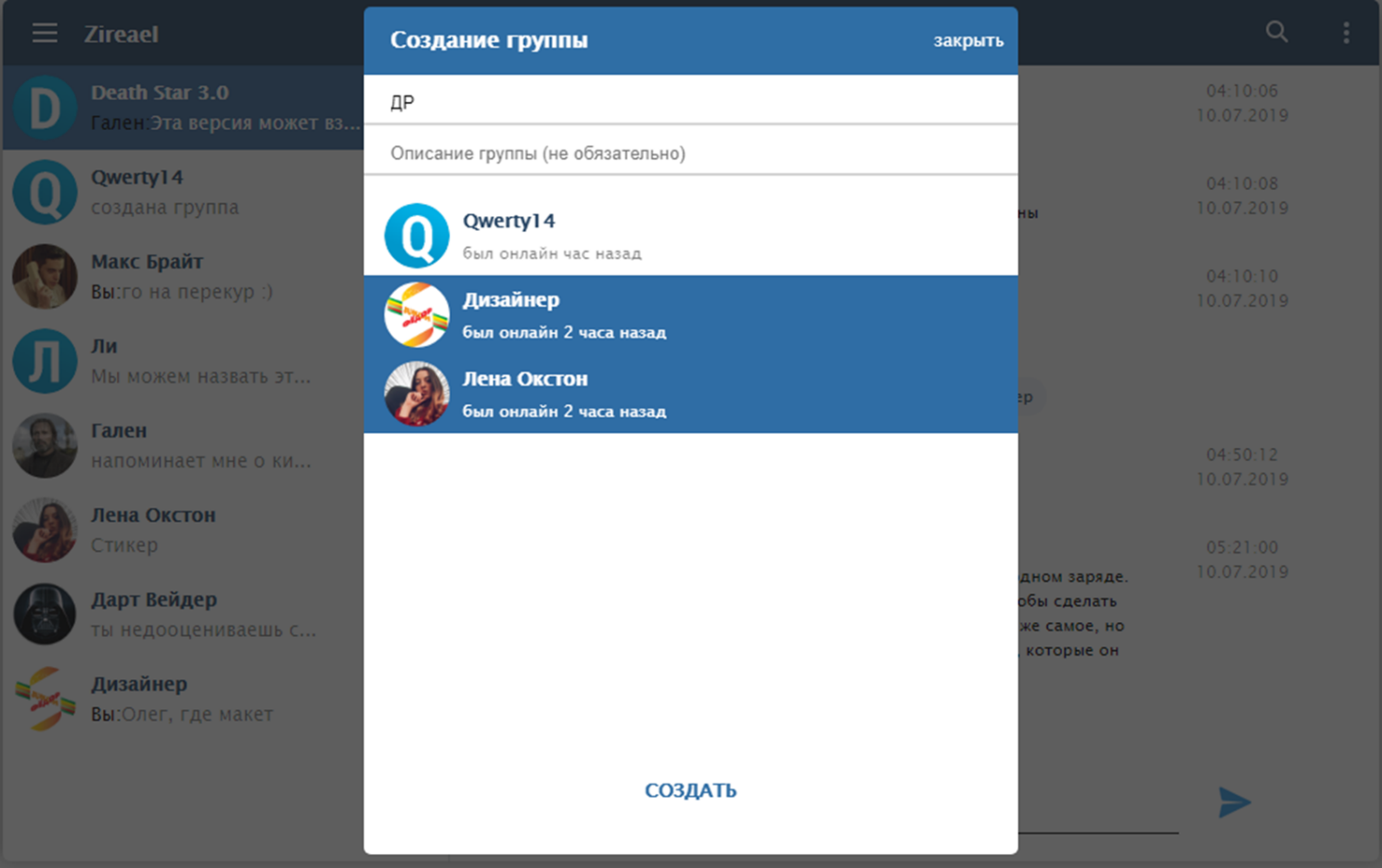
Выделение сообщений



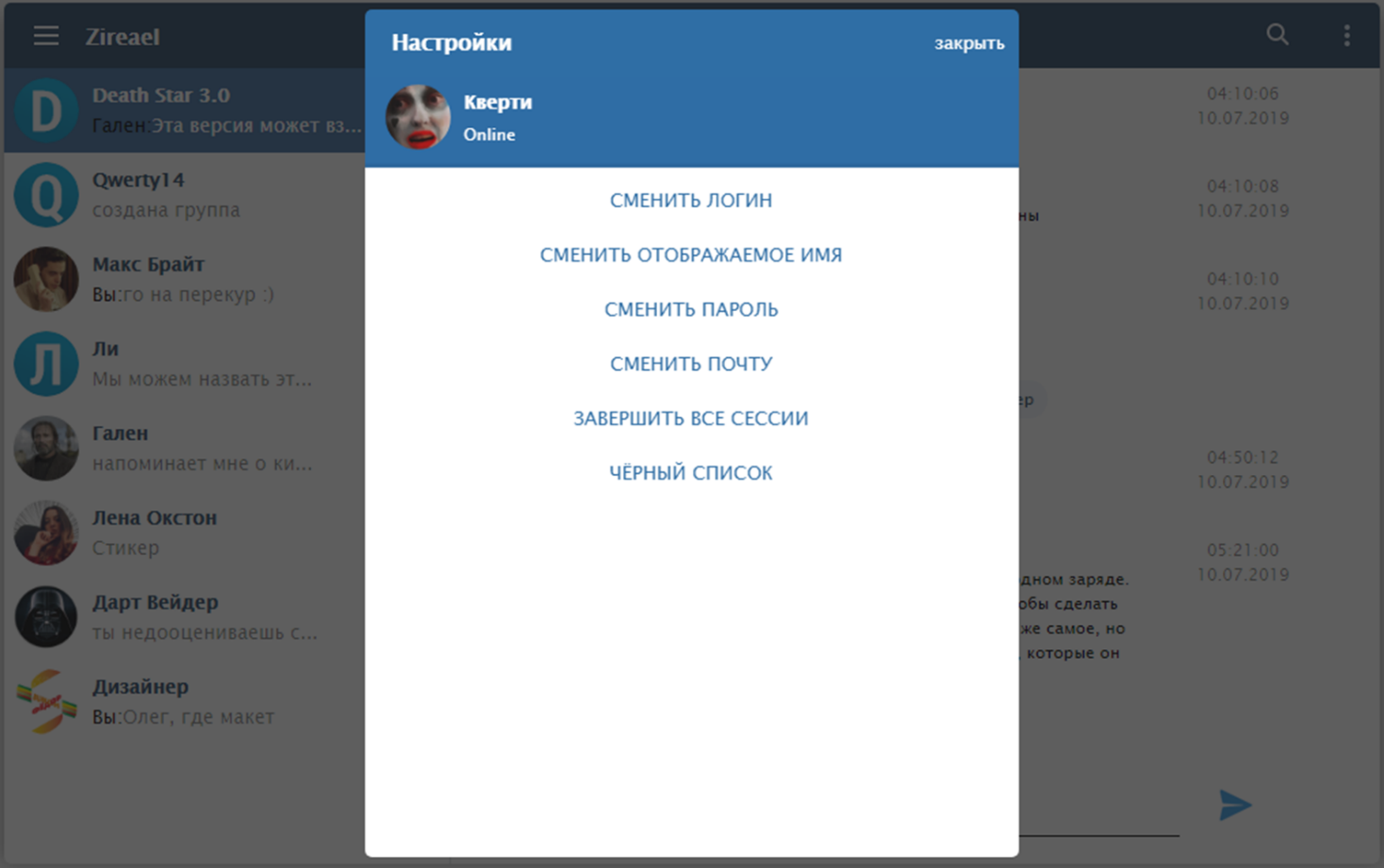
Контакты

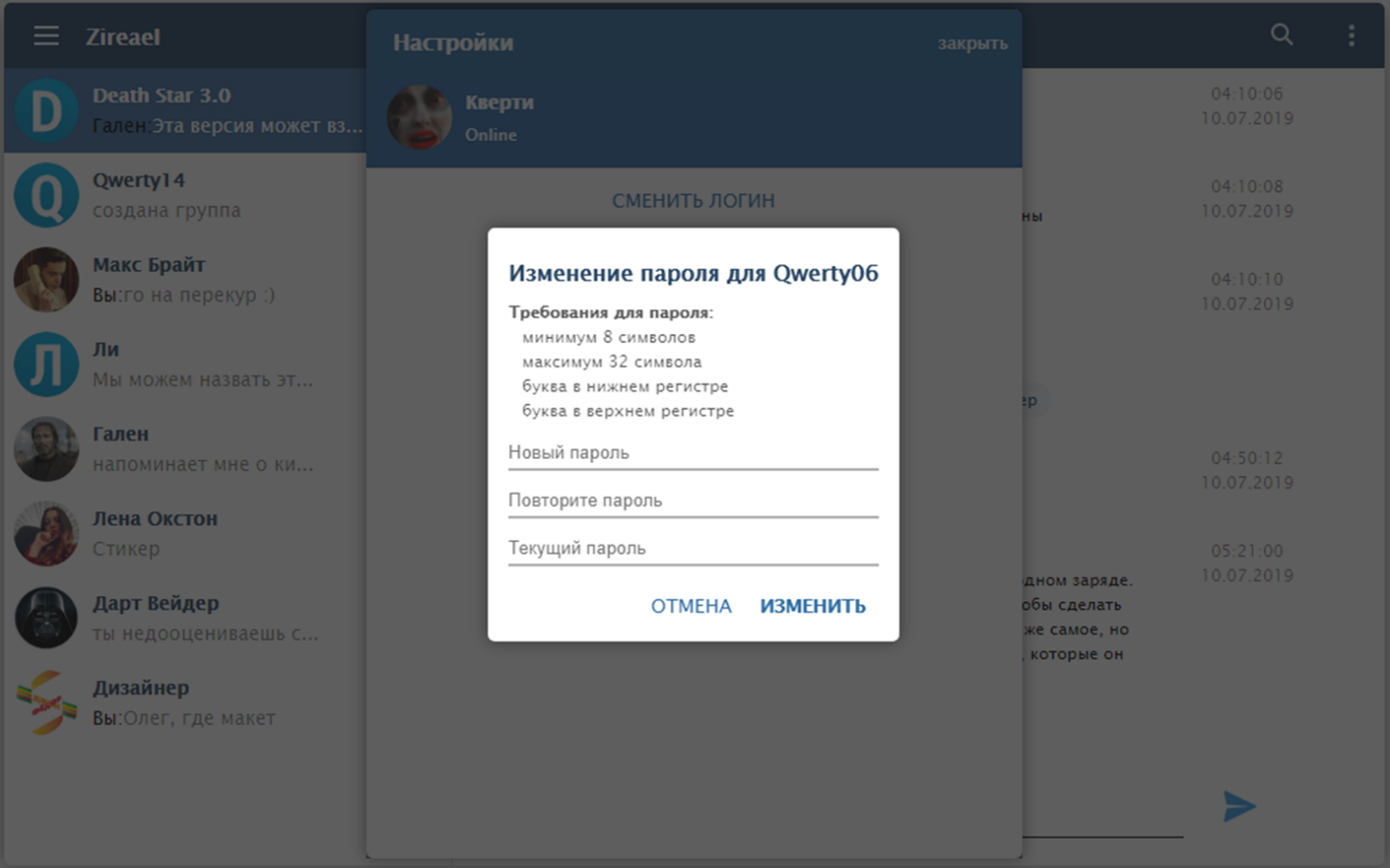


Создание группы

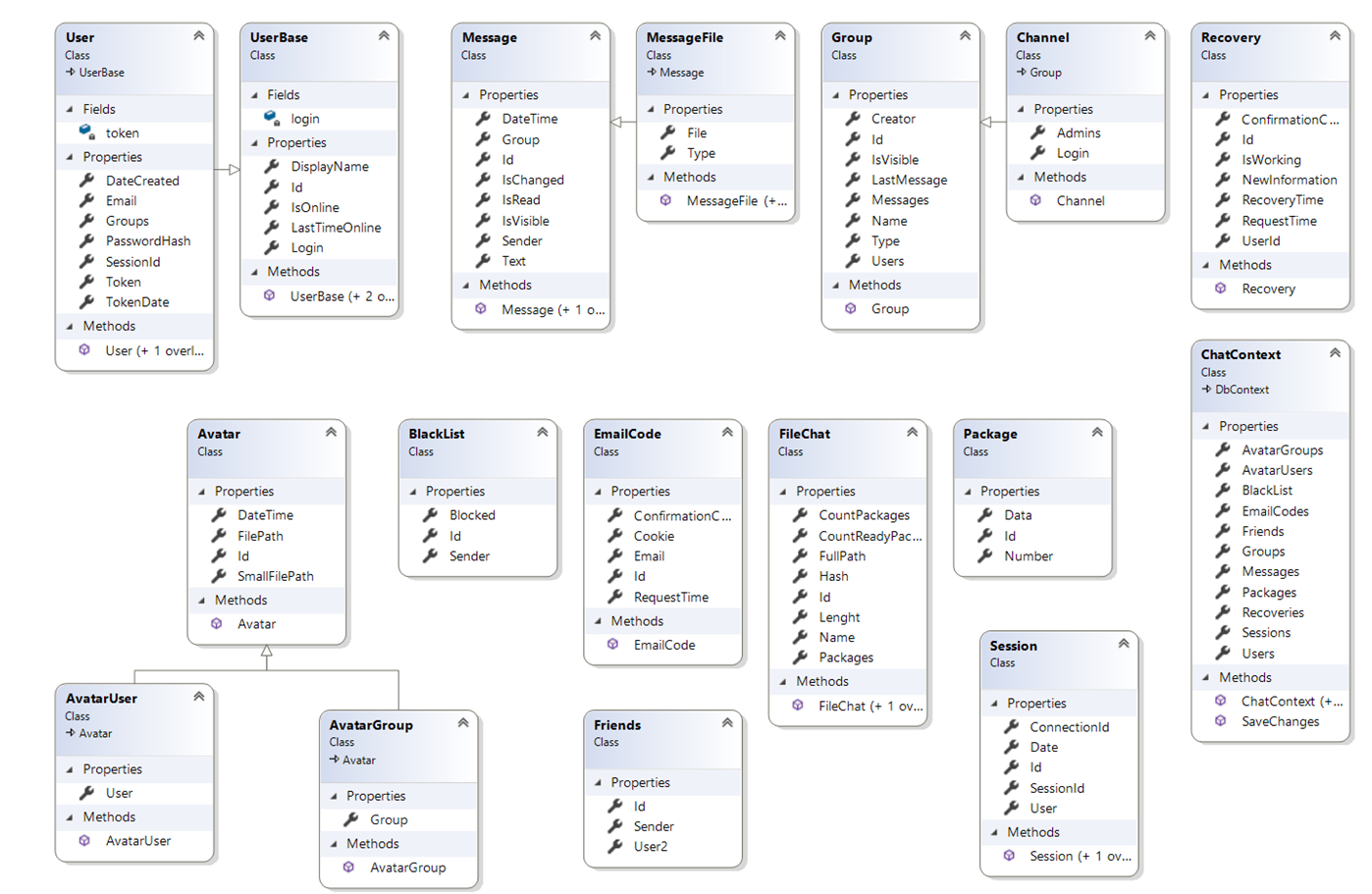


Настройки



Изменение пароля

# Дополнения

****