# Diagrama de Módulos

# Api

Módulo encargado de asociar funciones a endpoints http y convertir los resultados al formato de comunicación en red.

Este módulo incluye:

## usersController

| **NOMBRE** | **PARAMS** | **DESCRIPCIÓN** | **RESPUESTA** |
| --- | --- | --- | --- |
| */users/login* | UserVo : user | Valida que el usuario “user” exista y la contraseña sea correcta.  En “user” sólo son necesarios email y contraseña. | String : { Ok,  Usuario no existe,  Contraseña incorrecta} |
| */users/register* | UserVo : user | Registra una nueva cuenta con su perfil asociado. | Ok  Cuenta ya registrada. |
| */users/delete* | UserVo : user | Elimina la cuenta y el perfil del usuario | Ok  Usuario no existe |
| */users/edit* | UserVo : user | Edita el perfil del usuario con el mismo correo que “user” | ProfileVO  Error |
| */users/get* | email | Devuelve información sobre un perfil.  Falla si el perfil no es propio ni de alguien con quien tienes un match. | ProfileVO  Error |

## chats

| ***chatsController*** | | | |
| --- | --- | --- | --- |
| */chats/getChat* | ChatVO  (data vacio) | Devuelve los chats abiertos de un usuario | Lista de ChatVO  Error |
| ***messageHub*** | | | |
| *sendMessage* | Message | Envía un mensaje del chat. Devuelve true si se ha enviado correctamente y false si no. | bool |

## 

## matchesController

| ***Matches*** | | | |
| --- | --- | --- | --- |
| */matches/request* | UserVO | Apuntarse a la lista de espera de matches | Ok  Error |
| */matches/get* | UserVO | Devuelve una lista de Usuarios con los que el usuario está matcheado | IEnumerable  <UserVO> |
| */matches/delete* | MatchVO | Elimina el match indicado | Ok  Error |

## datesController

| ***Citas*** | | | |
| --- | --- | --- | --- |
| *getDefault* |  | Obtener un plan por defecto | PlanVO |

Sólo se muestra la función que se utiliza, el resto de documentación para cosas que no se llegaron a implementar se conserva en el [Anexo I](#_69maqwmp33xp).

## 

# AppLogic

## usersLogic

| **Función** | **Descripción** |
| --- | --- |
| int login(string email, string password) | Devuelve   * 0 si es correcto * 1 si las contraseñas no coinciden * 2 si el usuario no existe |
| int register(UserVo user) | Devuelve   * 0 si es correcto. * 1 si ya existe el usuario * 2 en el resto de casos |
| bool delete(string email, string password) | Devuelve   * Verdad si ha sido posible la eliminación de un usuario existente de la base de datos. * Falso en caso contrario. |
| bool edit(string email, string nombre, string apellidos, string sexo, DateTime fNacimiento, string localidad, string meGusta1, string meGusta2, string meGusta3, string noMeGusta1, string noMeGusta2, string noMeGusta3, string orientacion, int capacidad) | Devuelve   * Verdad si ha sido posible actualizar la información de “user” * Falso en caso contrario. |
| UserVO get(string email) | Devuelve   * un usuario VO con su perfil asociado. |

## chatsLogic

| **Función** | **Descripción** |
| --- | --- |
| ChatVO getChat(string emisor, string receptor) | Devuelve   * el ChatVO entre los usuarios emisor y receptor |
| bool saveMessage(Message m) | Devuelve   * Verdad si se ha podido guardar el mensaje * Falso en caso contrario |

## matchesLogic

| **Función** | **Descripción** |
| --- | --- |
| List<UserVO> getMatches(string email) | Devuelve   * La lista de usuarios con los que el usuario cuya clave es email esta emparejado. |
| bool deleteMatch(string email1, string email2) | Devuelve   * Verdad si se ha podido eliminar el match * Falso en caso contrario. |
| bool requestNewMatch(string email) | Devuelve   * Verdad si se ha añadido a la lista de espera de emparejamientos. * Falso en caso contrario. |

# 

# Model

## Dao

* **usersDAO**
  + UserVO getUser(string email)
  + bool createUser(UserVO usuario)
  + bool deleteUser(UserVO usuario)
  + bool editUser(UserVO usuario)
* **chatsDAO**
  + bool saveMessage(string sender, string reciever, string content, DateTime date)
  + List<Message> GetMessages(string sender, string reciever)
* **matchesDAO**
  + bool addMatch(string email1, string email2)
  + bool deleteMatch(string email1, string email2)
  + bool findMatch(string email1, string email2)
  + string getMatches(string email)

## Vo

* **UserVO**
  + email
  + password
  + nombre
  + apellidos
  + fNacimiento
  + sexo
  + localidad
  + meGusta(1-3)
  + noMeGusta(1-3)
  + foto
  + orientacion
  + capacidad
* **ChatVO**
  + emisor
  + receptor
  + data
* **Message**
  + sender
  + reciever
  + isSent
  + message
  + date
* **PlanVO**
  + lat
  + lon
  + donde
  + plan
* **DateVO**
  + email1
  + email2
  + momento
  + plan
* **MatchVO**
  + email1
  + email2

## DBManager

Módulo que gestiona las conexiones con la BD, ofrece el método:

* NpgsqlConnection getDBConnection()

que devuelve una conexión con la base de datos.

# IDEAS/PREGUNTAS

¿Cómo representamos los lugares? ¿Coordenadas? Hay bases de datos públicas de restaurantes para empezar, César sabe algo sobre el tema.

¿Especificamos códigos de respuesta?

<https://developer.mozilla.org/es/docs/Web/HTTP/Status>

¿Hora en formato unix timestamp?

# ANEXO I

Documentacion de la API relacionada con citas, no se ha llegado a implementar.

## 

| *suggestPlan* | email\_send  email\_recieve | Dados dos usuarios el sistema sugiere un plan y un lugar acorde con sus tags | DateVO  Error |
| --- | --- | --- | --- |
| *acceptPlan* | email\_send  email\_recieve | Confirma el plan y envía de email\_send a email\_recieive la confirmación | Ok  Error |
| *getDefault* | email\_send  email\_recieve | Pide un plan predeterminado al servidor | Ok  Error |
| *sendPlan* | email\_send  email\_recieve  cita | Un usuario envía una sugerencia de cita a otro | Ok  Error |
| *suggestFecha* | email\_send  email\_recieve  momento | Un usuario envía una sugerencia de fecha y hora a otro | Ok  Error |
| *acceptFecha* | email\_send  email\_recieve | Confirma la fecha y envía de email\_send a email\_recieive la confirmación. | Ok  Error |
| *checkStatus* | email1  email2 | Dados dos usuarios, devolver el estado del proceso de selección de cita, fecha y hora.  Ej: Si hay una sugerencia devolverla, sino decir que no hay nada sugerido, etc.. | cita por aceptar o rechazar  estado de la sugerencia  Error  etc… |