












Carpeta principal del programa:

 .git	03/06/2019 19:47	Carpeta de archivos	
 app	03/06/2019 19:46	Carpeta de archivos	
 config	03/06/2019 19:46	Carpeta de archivos	
 node_modules	03/06/2019 19:47	Carpeta de archivos	
 public	03/06/2019 19:47	Carpeta de archivos	
 views	03/06/2019 19:47	Carpeta de archivos	
 LICENSE	03/06/2019 19:46	Archivo	2 KB
 package.json	03/06/2019 19:47	Archivo JSON	1 KB
 package-lock.json	03/06/2019 19:47	Archivo JSON	109 KB
 README.md	03/06/2019 19:46	Archivo MD	4 KB
 server	03/06/2019 19:47	Archivo JavaScript	32 KB

.git: Arxius necessaris de Github.

app: Conté dues coses. Routes y models.

config: Configuració del passport. Aquí conté la config de autenticació, el access al MongoDB i l'arxiu de config del passport.

node_modules: Llibreries i arxius necessaris per el node

public: Carpeta que conté els scripts, imatges, etc necessaris per la web.

views: Vistes de l'aplicació. Aquí es guarden tots els .ejs (els arxius html web)

LICENSE: Licencia del PassportJS. Una llibreria utilitzada pel control d'usuaris

package.json:

package-lock.json:

README.md: Readme del projecte amb la informació del joc, com jugar, etc.

server.js: Servidor NodeJS que serveix la web i controla el funcionament de la web.

Carpetes desglossades:

APP

 models	03/06/2019 19:46	Carpeta de archivos	
 routes	03/06/2019 19:46	Archivo JavaScript	5 KI

Routes: Aquest és l'arxiu de configuració de les rutes de la nostra web. Utilitzem la llibreria Express de node per fer-ho funcionar.

Aquí definim la url, els paràmetres que passem i quina vista te que carregar.

També podem pasar la funció isLoggedIn, encarregada de controlar que el usuari estigui registrat. Si no ho està, no el deixa accedir-hi.

```

// show the home page (will also have our login links)
app.get('/', function(req, res) {
  res.render('index.ejs');
});

// perfil SECTION =====
app.get('/perfil', isLoggedIn, function(req, res) {
  res.render('perfil.ejs', {
    user : req.user
  });
});

// Jugar =====
app.get('/jugar', isLoggedIn, function(req, res) {
  res.render('jugar.ejs', {
    user : req.user
  });
  res.sendFile(__dirname + '/js/jugar.js');
});

app.get('/test', function(req, res) {
  res.render('test.ejs', {
    user : req.user
  });
});

// LOGOUT =====
app.get('/logout', function(req, res) {
  req.logout();
  res.redirect('/');
});

```

Models: Aquesta carpeta conté el fitxer de configuració de la taula usuaris al MongoDB que utilitza el PassportJS. Un cop registrem un usuari, omplirà aquest camps i el guardarà a la BD interna.

```

1 var mongoose = require('mongoose');
2 var bcrypt   = require('bcrypt-nodejs');
3
4
5 // define the schema for our user model
6 var userSchema = mongoose.Schema({
7
8   local    : {
9     email   : String,
10    usuario  : String,
11    password : String,
12    sexo     : String,
13    nivel    : {type: Number, default:0},
14    avatar   : String
15  }
16 });
17
18 // generating a hash
19 userSchema.methods.generateHash = function(password) {
20   return bcrypt.hashSync(password, bcrypt.genSaltSync(8), null);
21 };
22
23 // checking if password is valid
24 userSchema.methods.validPassword = function(password) {
25   return bcrypt.compareSync(password, this.local.password);
26 };
27
28 // create the model for users and expose it to our app
29 module.exports = mongoose.model('User', userSchema);
30

```

Config

 auth	03/06/2019 19:46	Archivo JavaScript	2 KB
 database	03/06/2019 19:46	Archivo JavaScript	1 KB
 passport	03/06/2019 19:46	Archivo JavaScript	7 KB

Es la carpeta de configuració del PassportJS, l'accés a MongoDB i la configuració de autenticació.

Auth: Fitxer de configuració de l'autenticació. Tenim, a part de l'accés local, access per FB, Twitter i Google. Actualment està desactivat per no controlar més coses de les necessàries.

```
1 // config/auth.js
2
3 // expose our config directly to our application using module.exports
4 /*
5 module.exports = {
6
7   'facebookAuth' : {
8     'clientID'       : 'your-secret-clientID-here', // your App ID
9     'clientSecret'   : 'your-client-secret-here', // your App Secret
10    'callbackURL'    : 'http://localhost:8080/auth/facebook/callback',
11    'profileURL'     : 'https://graph.facebook.com/v2.5/me?fields=first_name,last_name,email',
12    'profileFields'  : ['id', 'email', 'name'] // For requesting permissions from Facebook API
13  },
14 },
15
16   'twitterAuth' : {
17     'consumerKey'    : 'your-consumer-key-here',
18     'consumerSecret' : 'your-client-secret-here',
19     'callbackURL'    : 'http://localhost:8080/auth/twitter/callback'
20   },
21
22   'googleAuth' : {
23     'clientID'       : 'your-secret-clientID-here',
24     'clientSecret'   : 'your-client-secret-here',
25     'callbackURL'    : 'http://localhost:8080/auth/google/callback'
26   }
27 };
28 */
29
30
```

Database: Arxiu de configuració de l'accés al MongoDB amb les credencials.

```
// config/database.js
module.exports = {
  'url' : 'mongodb://localhost/passport' // looks like mongodb://<user>:<pass>@mongo.onmodulus.net:27017/Mikha4ot
};
```

Passport: Fitxer que conté i controla el access i registre d'usuaris. Es el fitxer base de la llibreria PassportJS.

- Login:

```
// =====
// LOCAL LOGIN =====
// =====
passport.use('local-login', new LocalStrategy({
  // by default, local strategy uses username and password, we will override with email
  usernameField : 'email',
  passwordField : 'password',
  passReqToCallback : true // allows us to pass in the req from our route (lets us check if a user is logged in or not)
}),
function(req, email, password, done) {
  if (email)
    email = email.toLowerCase(); // Use lower-case e-mails to avoid case-sensitive e-mail matching

  // asynchronous
  process.nextTick(function() {
    User.findOne({ 'local.email' : email }, function(err, user) {
      // if there are any errors, return the error
      if (err)
        return done(err);

      // if no user is found, return the message
      if (!user)
        return done(null, false, req.flash('loginMessage', 'No user found.'));

      if (!user.validPassword(password))
        return done(null, false, req.flash('loginMessage', 'Oops! Wrong password.'));

      // all is well, return user
      else
        return done(null, user);
    });
  });
});
});
```

- Registre:

```
passport.use('local-signup', new LocalStrategy({
  // by default, local strategy uses username and password, we will override with email
  usernameField : 'email',
  passwordField : 'password',
  passReqToCallback : true // allows us to pass in the req from our route (lets us check if a user is logged in or not)
}),
function(req, email, password, done) {
  if (email)
    email = email.toLowerCase(); // Use lower-case e-mails to avoid case-sensitive e-mail matching

  // asynchronous
  process.nextTick(function() {

    // if the user is not already logged in:
    if (!req.user) {
      User.findOne({ 'local.email' : email }, function(err, user) {
        // if there are any errors, return the error
        if (err)
          return done(err);

        // check to see if theres already a user with that email
        if (user) {
          return done(null, false, req.flash('signupMessage', 'Este email ya existe.'));
        } else {

          User.findOne({'local.usuario' : req.param('usuario')}, function(err, user){
            if(err)
              return done(err);
            if(user){
              return done(null,false, req.flash('signupMessage', "Este usuario ya existe"));
            }
            else {
              // create the user
              var newUser = new User();
              newUser.local.nivel = 1;
              newUser.local.usuario= req.param('usuario');
              newUser.local.avatar = null;
              newUser.local.email = email;
              newUser.local.sexo = req.param('gender');
            }
          });
        }
      });
    }
  });
});
```

Public

img	03/06/2019 19:47	Carpeta de archivos
js	03/06/2019 19:47	Carpeta de archivos

Carpeta amb les imatges i scripts necessaris per la web.

img: Aquesta carpeta conté totes les imatges de la web, imatges d'alertes, de accions de rols, fons de pantalla....



js: Scripts i accés al Firebase

firebase	03/06/2019 19:47	Archivo JavaScript	1 KB
jugar	03/06/2019 19:47	Archivo JavaScript	23 KB
key.json	03/06/2019 19:47	Archivo JSON	3 KB

firebase: Arxiu amb la configuració d'accés al Firebase i la variable "db" que utilitzem als nostres codis.

```
// Initialize Firebase
var config = {
  apiKey: "AIzaSyDvlVoN5isTaupv0tnoL9m8UzPEdHT3qXg",
  authDomain: "werewolf-140eb.firebaseio.com",
  databaseURL: "https://werewolf-140eb.firebaseio.com",
  projectId: "werewolf-140eb",
  storageBucket: "werewolf-140eb.appspot.com",
  messagingSenderId: "352508924475"
};
firebase.initializeApp(config);
const db = firebase.firestore();
```

key.json: Key personal del projecte necessària per la connexió del NodeJS amb el Firebase.

```
{
  "type": "service_account",
  "project_id": "werewolf-140eb",
  "private_key_id": "3ff60ccc5b0710771ae251ef9c6aa4673ac1f47",
  "private_key": "-----BEGIN PRIVATE KEY-----nMIIIEvQIBADANBgkqhkiG9w0BAQEFAAQCAQAwgAgEAAoIBAQCsuRzu04FrAIAa\nTXM9cmn7clFD2KXORk+r2/gFVqFJcTpygTnV1jP1sKCA7UEq5RjH4fwHS4C2jFaY\nngvzqx33X3H2fUVHah6oYij/hkqBY8ZaGYc5wSIjVq7+twlQx17\n",
  "client_email": "firebase-adminsdk-3rcv1@werewolf-140eb.iam.gserviceaccount.com",
  "client_id": "110912380626798305483",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-3rcv1@werewolf-140eb.iam.gserviceaccount.com"
}
```

Jugar.js: Es l'arxiu base al nostre joc per a la part de client. Un cop accedim a la pestanya "Jugar" començarà a fer la connexió amb Sockets i comprovarà l'usuari.


```

//===== FUNCIONES SOCKET =====

//Conexión al servidor
var socket = io.connect('http://localhost:8080', { 'forceNew': true });
//

// HA ENTRADO UN JUGADOR
socket.on('hola', function(EstadoPartida) {
  console.log("El servidor ha recibido al usuario "+userId);
  //EL ESTADO DE LA PARTIDA?
  console.log("El estado de la partida es: "+EstadoPartida);
  //SI LA PARTIDA ESTÁ SIN EMPEZAR, LE DEJAMOS ENTRAR
  if(EstadoPartida=="Pendiente"){
    tablero();
    //EXISTIA?
    //SI NO EXISTE LO AÑADIRÁ, SINO NO
    comprobar_usuario(userId, username, IDPartida);
  }
  else {
    check_usuario_sala(userId, IDPartida);
  }
})

```

També tenim definides totes les variables necessàries per al funcionament del joc (part client)

```

//=====VARIABLES=====
var IDPartida= "avKFrF5ZFS9OxrJDgAy3";
var estado="";

//Datos de usuario
var rol = null;
var partidaAcabada=false;
var UsuarioVotado = "";
var VotacionesRechas=false;
var VotacionesRechasLobo= false;
let MasVotos=0;
let MasVotado=null;
let MasVotosLobos=0;
let MasVotadoLobos=null;
var avisoMuerte=false;
var Muerte = false;
var MiEquipo=null;
var accion = false;
var accionVidente = false;
var rolvisto="";
var ArrayLobosCliente = [];
var asesinado;
var IdBufon=null;
var Ganador=false;
var EquipoGanador=null;

console.log("===DATOS===");
const userId= $('#userid').text();
console.log("id: "+userId);
var username= $('#username').text();
console.log("usuario: "+username);
var genero= $('#sexo').text();
console.log("Genero: "+genero);
//comprobamos que el usuario no esté muerto
comprobarMuerte();
console.log("Estado:"+Muerte);

```

Controla el ciclo de nit/día/votació de la part client. Sempre ho comprova amb el servidor per evitar un “desfase” o tramples al actualitzar o tancar.

```

estado = EstadoPartida;
//Comprobamos si estamos muertos
comprobarMuerte();
$('#ContadorVotos').hide();
var Info = document.getElementById('InfoPartida');
//actualizamos tablero por si hay cambios
tablero();

//Comprobamos los roles en todos los casos menos en empezar, que es cuando se asignan
if(EstadoPartida!="Empezada")
    coger_rol();

console.log("¡LA PARTIDA HA TENIDO UN CAMBIO DE ESTADO!");
console.log(EstadoPartida);

if(EstadoPartida=="Empezada"){
    //contador para empezar la partida. Le pasamos el siguiente estado
    console.log("Cuenta atrás para empezar la partida. aprox:"+tiempo+" segundos");
    Info.innerHTML = "Cuenta atrás para empezar asignar los roles y comenzar. <div id='segundos'><br>Tiempo: 0 segundos</div>";
}

if(EstadoPartida=="Asignando"){
    //contador para empezar la partida. Le pasamos el siguiente estado
    console.log("Asignando roles...");
    Info.innerHTML = "Cuenta atrás para empezar la partida. <div id='segundos'><br>Tiempo: 0 segundos</div>";
    //cargar_accion();
}

if(EstadoPartida=="Noche"){
    //si se ha votado durante el día
    if(VotacionesHechas==true){
        //reiniciamos la var
        VotacionesHechas=false;
        //Enviamos al servidor el usuario que ha sido mas votado
        socket.emit("MasVotado", MasVotado, MasVotos);
        //Comprobamos si somos nosotros quienes hemos muerto y si no hemos avisado antes
        if(MasVotado==userId && avisoMuerte == false && partidaAcabada==false && rol!="Bufón"){
            Muerte = true;
            avisoMuerte=true;
            avisoDeMuerte();
        }
    }

    //activamos la habilidad de la vidente
    accionVidente=true;
    //logs
    console.log("Es de noche.");
    console.log("Los lobos votan a un aldeano para morir");

    //Cambiamos la apariencia de la aldea a noche
    $("body").attr('class', 'noche');
    Info.innerHTML = ("Es de noche. Los lobos votan a un aldeano para morir. <div id='segundos'><br>Tiempo: 0 segundos</div>");

    //Cambiamos el chat al de los lobos (si lo somos)
}

```

Controla els sockets que envia el servidor al client i executa la funció de cadascun d'ells.

Exemples:

```

socket.on("ActualizarTablero", function(){
    //El servidor pide actualizar el tablero
    tablero();
});

socket.on("rolesAsignados", function(){
    //cogemos el rol del usuario ahora por si se ha conectado tarde
    coger_rol();
});

socket.on('tiempo', function(segundos) {
    var TiempoPartida = document.getElementById('segundos');

    console.log(segundos);
    TiempoPartida.innerHTML = " <br>Tiempo:</b><i> "+segundos+" segundos</i>";
});

//El servidor ha recibido un voto
socket.on('VotoRecibido', function(ArrayVotos){
    let result=contarFreq(ArrayVotos);
    //Ponemos los contadores a 0
    $('span').text("0");




    for (var i = 0; i < result.length; i++) {
        var UsuarioVotado= result[0][i];
        console.log("EL ID" + result[0][i]);
        console.log("NUMERO DE VOTOS" + result[1][i]);
        var NumVotos = result[1][i];
        var selecSpan = eliminarEspacios(UsuarioVotado);
        $(''+selecSpan).text(NumVotos);
        if(NumVotos>MasVotos){
            MasVotos= NumVotos;
            MasVotado = UsuarioVotado;
        }
    }
}

```

També conté les alertes amb SweetAlert, funcions necessaries per a les accions de rol, per al desenvolupament de la partida i actualitzacions a la BD de Firebase.

```
function comprobarMuerte() {
  db.collection("usuarios").where("id_usuario", "=", userId)
    .get().then(function(querySnapshot) {
      if(querySnapshot.size > 0){
        querySnapshot.forEach(function(doc) {
          if(doc.data().estado=="muerto"){
            Muerte=true;
          }
        });
      }
    });
  sleep(900);
  if(MasVotado==userId && avisoMuerte == false ){
    if(rol == "Bufón"){
      alertHasGanado();
    }
    else{
      Muerte = true;
      avisoMuerte=true;
      avisoDeMuerte();
    }
  }
}
```

views: Directori amb les vistes de la web. Cada una de les vistes es un apartat.

 connect-local.ejs	03/06/2019 19:47	Archivo EJS	2 KB
 index.ejs	03/06/2019 19:47	Archivo EJS	1 KB
 jugar.ejs	03/06/2019 19:47	Archivo EJS	11 KB
 login.ejs	03/06/2019 19:47	Archivo EJS	2 KB
 perfil.ejs	03/06/2019 19:47	Archivo EJS	2 KB
 signup.ejs	03/06/2019 19:47	Archivo EJS	3 KB

connect-local.ejs: Arxiu al que s'accedeix si intentes mouret sense registre.

index.ejs: Pàgina principal de la web.

jugar.ejs: Pàgina de joc. Conté el taulell de joc, el xat i la part visual de la partida.

login.ejs: Login web.

perfil.ejs: Perfil provisional de l'usuari on es pot veure la informació

signup.ejs: Registre web.

License

Llicència de la llibreria PassportJS i la seva estructura web.

The MIT License (MIT)

Copyright (c) 2014 Scotch.io

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Package.json

Informació json de l'aplicació NodeJS

```
{
  "name": "Werewolfs vs Village",
  "main": "server.js",
  "dependencies": {
    "bcrypt-nodejs": "latest",
    "body-parser": "~1.15.2",
    "connect-flash": "~0.1.1",
    "cookie-parser": "~1.4.3",
    "ejs": "~2.5.2",
    "express": "~4.14.0",
    "express-session": "~1.14.1",
    "firebase-admin": "^7.3.0",
    "method-override": "~2.3.6",
    "mongodb": "^3.2.3",
    "mongoose": "~4.13.1",
    "morgan": "~1.7.0",
    "passport": "~0.3.2",
    "passport-facebook": "~2.1.1",
    "passport-google-oauth": "~1.0.0",
    "passport-local": "~1.0.0",
    "passport-twitter": "~1.0.4",
    "socket.io": "^2.2.0",
    "sweetalert2": "^8.10.7"
  }
}
```

Package-lock.json

Fitxer amb la informació de les llibreries que utilitza el NodeJS

```
"name": "Werewolves vs Village",
"requires": true,
"lockfileVersion": 1,
"dependencies": {
  "@firebase/app": {
    "version": "0.3.17",
    "resolved": "https://registry.npmjs.org/@firebase/app/-/app-0.3.17.tgz",
    "integrity": "sha512-/8lDeeIkgdCIMfzrFBQ3bcdSkF8bx4KCP8pKMP0G/HYKoeM8I9eF4zLxL5ABzRjvodhK9K0YOn0jRrNrGD9g==",
    "requires": {
      "@firebase/app-types": "0.3.10",
      "@firebase/util": "0.2.14",
      "dom-storage": "2.1.0",
      "tslib": "1.9.3",
      "xmlhttprequest": "1.8.0"
    }
  },
  "@firebase/app-types": {
    "version": "0.3.10",
    "resolved": "https://registry.npmjs.org/@firebase/app-types/-/app-types-0.3.10.tgz",
    "integrity": "sha512-1+5BJtSQopalBK1Y/YuSaB9KF9PmDj37FLV0Sx3qJh5B3IthCuZbPclVpbbbae/Q2gudl0G212BBsUMGHP+fQ==",
    "requires": {}
  },
  "@firebase/database": {
    "version": "0.3.20",
    "resolved": "https://registry.npmjs.org/@firebase/database/-/database-0.3.20.tgz",
    "integrity": "sha512-fZHRlRlQ1ND/UrzI1beUTRkftjMvMEiUOar6y1FZq0j2KNV04GrF95UGqL0HBSHzz1BKzaDYAcJze2D6C4+Q==",
    "requires": {
      "@firebase/database-types": "0.3.11",
      "@firebase/logger": "0.1.13",
      "@firebase/util": "0.2.14",
      "faye-websocket": "0.11.1",
      "tslib": "1.9.3"
    }
  },
  "@firebase/database-types": {
    "version": "0.3.11",
    "resolved": "https://registry.npmjs.org/@firebase/database-types/-/database-types-0.3.11.tgz",
    "integrity": "sha512-iRAZzs7Z1mmvh7x0XLR1MAO616mlHjW9mlYcfJ6E/8+zItHnbVH4iiVVKC39r1wMGrtPMz8FiUWoasPF5dA==",
    "requires": {}
  },
  "@firebase/logger": {
    "version": "0.1.13",
    "resolved": "https://registry.npmjs.org/@firebase/logger/-/logger-0.1.13.tgz",
    "integrity": "sha512-wIbLwQ2oJCVhIE7J3FDxpScKY84f5ctEEj0i0PB+Yn2dN8AwqtM7Yf8rtcY8cxntv8dyR+i7GNq1Nd89cGxkA==",
    "requires": {}
  },
  "@firebase/util": {
    "version": "0.2.14",
    "resolved": "https://registry.npmjs.org/@firebase/util/-/util-0.2.14.tgz",
    "integrity": "sha512-2keILraORST+SucCNWrt/IB2zI/IzumHHYm9aqzW91J3XURiWmBHAYrvaPVP7///gDhJAo+NNDUCAJH/Y4PmvA==",
    "requires": {
      "tslib": "1.9.3"
    }
  },
  "@google-cloud/common": {
    "version": "0.32.1",
    "resolved": "https://registry.npmjs.org/@google-cloud/common/-/common-0.32.1.tgz",
    "integrity": "sha512-bLdPzFvvBDMcVkwsoBtygE9oUm3yzNmFa71gvGugcII/GqvNP2tb6RiYsDHPq98kvignhcgHGD15wyNgxaCo8bKQ==",
    "optional": true,
    "requires": {}
  }
}
```

Readme.md

Readme amb els textos de que es el “Lobo” amb els que ens hem basat, com es el nostre joc, com es desenvolupa una partida, com guanyar i els nostres noms.

```
# Werewolves VS Village

Nuestro proyecto final se basa en el juego de cartas "los hombres lobo de Castronegro", un juego de cartas ideal para reir con los amigos y pasar la tarde, llevado a los navegadores.

## Juego original

Los hombres lobo (Les Loups-Garous de Thiercelieux en su edición original francesa) es un juego de cartas de 8 a 18 jugadores creado en 2001 por Dmitry Davidoff, Philippe des Pallières y Hervé Marly e inspirado en el juego precedent

# Objetivo del juego

En general, existen dos bandos en la partida: Aldeanos y Hombres Lobo. El objetivo de cada equipo es eliminar a todos los miembros del otro, con la particularidad de que los aldeanos (la mayoría) ignoran quiénes son los hombres lobo

El juego se articula en torno a dos fases: fase de Día (todos los jugadores tienen los ojos abiertos y hablan, tratando de encontrar a los hombres lobo, mientras los hombres lobo se hace pasar por aldeanos) y fase de Noche (todos los

Durante la fase de la Noche los hombres lobo abrirán los ojos en su turno para elegir una víctima, que morirá al día siguiente. Es posible que existan otros turnos por la noche, durante los cuales únicamente abrirá los ojos un aldeano

Las fases de noche y día se alternan sucesivamente hasta que únicamente queden hombres lobo o aldeanos vivos, y la partida termine con la victoria de uno de los dos bandos (es posible que la partida termine en empate si se juega con

# Nuestro juego

Una vez registrados/logueados podremos acceder a la sala de juego donde, después de juntar de 8 a 16 jugadores, la partida empezará.

El ciclo empezará en noche. Los hombres lobo y el asesino acabaran con sus objetivos mientras la aldea duerma. A la llegada del día, dos habrán muerto si así lo han querido y la aldea no ha hecho nada para evitarlo.

Por último, llega la fase de votaciones. En esta fase la aldea decide quien debe morir.

El ciclo sigue así hasta que alguno de los equipos gana.

# Cómo ganar

La partida acabará si:

- La aldea acaba con el asesino y los lobos
- Los lobos son mayoría y han matado al asesino
- Si el asesino ha matado a todo el mundo
- Si es 1 lobo y 1 asesino
- Si es 1 lobo y un aldeano
- Si es 1 asesino y 1 aldeano

### Creadores

Oriol Soler y Cristina Ferrer
```

Server.js

Arxiu base de la nostra web i de la part servidor de la partida.

Al començament importa totes les llibreries necessàries per funcionar correctament i configura/exporta la configuració del Mongo i el Express.

També fa un path directe a la carpeta img i js per poder-ho importar molt més fàcil.

```
// set up =====
// get all the tools we need
var express = require('express');
var app = express();
var port = process.env.PORT || 8080;
var mongoose = require('mongoose');
var passport = require('passport');
var flash = require('connect-flash');

var morgan = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var session = require('express-session');

var configDB = require('./config/database.js');

//definir archivos
var path = require("path");
app.use('/js', express.static(__dirname + '/public/js'));
app.use('/img', express.static(__dirname + '/public/img'));

// configuration =====
mongoose.connect(configDB.url); // connect to our database

require('./config/passport')(passport); // pass passport for configuration

// set up our express application
app.use(morgan('dev')); // Log every request to the console
app.use(cookieParser()); // read cookies (needed for auth)
app.use(bodyParser.json()); // get information from html forms
app.use(bodyParser.urlencoded({ extended: true }));

app.set('view engine', 'ejs'); // set up ejs for templating

// required for passport
app.use(session({
  secret: 'ilovescotchscotchyscotchscotch', // session secret
  resave: true,
  saveUninitialized: true
}));
app.use(passport.initialize());
app.use(passport.session()); // persistent login sessions
app.use(flash()); // use connect-flash for flash messages stored in session

// routes =====
require('./app/routes.js')(app, passport); // load our routes and pass in our app and fully configured passport
```

Declara les variables necessàries per a la connexió al Firebase, exporta la Key.json personal del projecte, declara variables de la partida...i seguidament controla els snapshots

fets a la taula partida i usuarios.

```
53 //CONEXIO DB
54 const admin = require('firebase-admin');
55
56 var serviceAccount = require('./public/js/key.json');
57
58 admin.initializeApp({
59   credential: admin.credential.cert(serviceAccount)
60 });
61
62 var db = admin.firestore()
63
64 //VARS DE LA DB
65 var DBJugadores = db.collection('usuarios');
66 var DBPartida = db.collection('partida');
67
68 //VARS DE LA PARTIDA
69 var IDPartida= "avKFrF5ZF590xrJDgAy3";
70 var IDSala = "phk5QBx6nefQH8rePDAz";
71 var EstadoPartida ="Pendiente";
72 var NumUsuarios=" ";
73 console.log("El estado actual de la partida es:" +EstadoPartida);
74
```

```
//SOCKET
var server = require('http').Server(app);
var io = require('socket.io')(server);
//chat
var messages = [];
var messagesN = [];
//votos
var votos = new Map();
var votosPiscopata = new Map();
var votosLobo = new Map();
var ArrayVotos = [];
var ArrayVotosLobo = [];
var ArrayMuertos= [];
var AuxNombreMasVotado;
var CopiaAuxMasVotado;
var CopiaAuxMasVotadoLobos;
//partida
var partidaAcabada = false;
//roles
var ArrayAldea = [];
var ArrayLobos = [];
var ObjetivoRol=null;
var NombreUsuarioRol=null;
var BalasRestantes=2;
var CargaHechicero=true;
var AguaBendita=1;
var cuchillada=true;
var IdBufon=null;
var ProtegidoDoctor=null;
var ProtegidoGuarda=null;
var Guardaespaldas=null;
var IdAsesino=null;
```


Després d'això, controla els Sockets. Executa les funcions quan rep un Socket o envia una funció amb un emit.

Tots els cicles estan connectats amb Sockets, igual que qualsevol canvi en la partida com la mort d'un usuari o un missatge al xat.

```
//====Socket====
io.on('connection', function(socket) {
  socket.emit('hola', EstadoPartida);
  io.sockets.emit("ActualizarTablero");

  //chat dia
  socket.emit('messages', messages);

  socket.on('new-message', function(data) {

    messages.push(data);

    io.sockets.emit('messages', messages);
  });
  //chat noche
  socket.emit('messagesN', messagesN);

  socket.on('new-messageN', function(data) {

    messagesN.push(data);

    io.sockets.emit('messagesN', messagesN);
  });

  //=====Roles=====
  //Lobo recibido en el cliente, avisa al servidor
  socket.on("EnviarLobo", function(userId){
    //Hemos recibido la id de un lobo, la guardamos en el array
    if(ArrayLobos.includes(userId))
      ArrayLobos.push(userId);
  });

  //Bufon
  //Bufón recibido en el cliente, avisa al servidor
  socket.on("EnviarBufon", function(userId){
    //Aguardamos el Bufón
    IdBufon=userId;
    //Lo enviamos a los clientes
    io.sockets.emit("DevolverBufon", IdBufon)
  });
  //ALDEA
  socket.on("EnviarAldea", function(aldeano){
    if(!ArrayAldea.includes(aldeano))
      ArrayAldea.push(aldeano);
  });
  //Psicopata
  socket.on("EnviarPsicopata", function(asesino){
    IdAsesino=asesino;
  });

  //pistolero
  socket.on("Balas", function(userId, UsuarioVotado, username){
    //el pistolero aun tiene balas?
    //...
  });
});
```


A part de totes les funcions amb sockets també té funcions JS per al funcionament de la partida, funcions per treballar canvis al Firebase o funcions auxiliars (com contadors, esperes de temps, càlculs, etc..)

```
function sleep(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}

function revelarRol(Usuario){
  db.collection("usuarios").where("id_usuario", "==", Usuario)
    .get()
    .then(function(querySnapshot) {
      querySnapshot.forEach(function(doc) {
        //Mostramos su rol cuando muere
        var rolvisible = doc.data().rol;
        // Lo matamos
        db.collection("usuarios").doc(doc.id).update({rol_visible: rolvisible});
      });
    });
  setTimeout(function() {
    console.log('Tiempo de espera por seguridad (revelar un rol)');
  }, 200);
}

function MatarUsuario(Usuario){
  db.collection("usuarios").where("id_usuario", "==", Usuario)
    .get()
    .then(function(querySnapshot) {
      querySnapshot.forEach(function(doc) {
        console.log(doc.id, " => ", doc.data());
        var rolvisible = doc.data().rol;

        // Build doc ref from doc.id
        db.collection("usuarios").doc(doc.id).update({estado: "muerto", rol_visible: rolvisible});
      });
    });

  setTimeout(function() {
    console.log('Tiempo de espera por seguridad (Matar usuario)');
  }, 200);
}

function sacarNombre(id){
  //Sacamos el nombre de usuario en firebase a partir de la id
  var AuxNombreUsuarioVotado;
  db.collection("usuarios").where("id_usuario", "==", id)
    .get()

```

La funció principal (tant per servidor com a client) pel funcionament de la partida, es la funció de “manejar_estado”. Cada X segons envia un canvi d'estat del següent part del cicle. Després s'encarrega de fer funcionar les funcions de cadascun d'aquest cicles.

```

//si hoy mas de 10, 10s
else if (NumUsuarios>10)
    tiempo_espera = 10000;

console.log("Hay suficientes jugadores para empezar, vamos a cambiar el estado de la partida dentro de "+tiempo_espera/1000+" segundos. Esperamos posibles nuevos jugadores.");
setTimeout(cambiar_estado,tiempo_espera,"Empezada");
}
if(EstadoPartida=="Empezada"){
    //asignar roles
    asignar_rols();
    //contador para empezar la partida, le pasamos el siguiente estado
    console.log("Cuenta atrás para empezar la partida: ");
    tiempo_espera=8;
    contador(tiempo_espera, "Asignando");
}

if(EstadoPartida=="Asignando"){
    tiempo_espera=3;
    contador(tiempo_espera, "Noche");
}

if(EstadoPartida=="Noche"){
    //Reiniciamos la cuchillada para el psicópata
    cuchillada=true;

    console.log("Es de noche.");
    console.log("Los lobos votan a un aldeano para morir");
    tiempo_espera=10;

    contador(tiempo_espera, "Día");
}

if (EstadoPartida=="Votaciones") {
    console.log("Es momento de votar a los lobos/ Psicópata");
    console.log("Volveré la noche");
    tiempo_espera=10;

    contador(tiempo_espera, "Noche");
}

if (EstadoPartida=="Día") {
    //Un nuevo día, reiniciamos el array de votos
    ArrayVotos=[];
    console.log("Votos reiniciados");
    console.log("Es de día.");
    console.log("Un par de aldeanos han muerto por el Psicópata y por los lobos");
    console.log("Es momento de discutir");
    tiempo_espera=10;

    contador(tiempo_espera, "Votaciones");
}
}
}

```

Un altre funció important és la funció d'assignar rol de forma aleatoria. Un cop la partida ha començat (s'ha tancat la sala) es començaran a repartir els rols.

Aquesta funció s'encarrega de repartir els valors d'un array o d'un altre segons el número de jugadors per a que cadascun tingui el seu rol. Segons el num de jugadors la partida será d'una manera o d'una altre. No son iguals les partides ni els rols.

```

else if (NumUsuarios==14) {
    Roles = ["Lobo", "Lobo", "Lobo", "Lobo", "Vidente", "Vidente", "Pistolero", "Psicópata", "Cura", "Doctor", "Bufón", "Guardaespalidas", "Hechicero", "Aldeano"];
    shuffle_rols(Roles);
}

else if (NumUsuarios==15) {
    Roles = ["Lobo", "Lobo", "Lobo", "Lobo", "Vidente", "Vidente", "Pistolero", "Psicópata", "Cura", "Doctor", "Bufón", "Guardaespalidas", "Hechicero", "Aldeano", "Aldeano"];
    shuffle_rols(Roles);
}

else if (NumUsuarios==16) {
    Roles = ["Lobo", "Lobo", "Lobo", "Lobo", "Vidente", "Vidente", "Pistolero", "Psicópata", "Cura", "Doctor", "Bufón", "Guardaespalidas", "Aldeano", "Hechicero", "Aldeano", "Aldeano"];
    shuffle_rols(Roles);
}

//error

```

Servir la web

```

// Launch *****
server.listen(8080, function() {
    console.log("Servidor corriendo en http://localhost:8080");
});

```