INTRODUCTION

Today, the use of technological tools is everywhere, they are used for anything, from communicating with loved ones in distant places, to looking for a place to eat. Among these tools are applications for mobile devices, commonly known as “apps”, these are of great help because just by installing it you already have access to it at any time you need.

But in this time where a large part of the interactions between people occurs through social networks, there are moments where remembering simpler times is provided, where you lived with your friends in parks, talking and playing soccer, where you only needed one court, a ball, and small teams to make matches.

Currently, these challenges can still be achieved, but that tradition has been lost over time, lending itself to occasions when one wants to participate in them, but does not know where they are currently playing, which makes it very difficult.

BACKGROUND

This project focuses on the development of a mobile application for the Android operating system, using Visual Studio Code software as a development environment and storing information in a database in the Firebase service.

Android is the most famous operating system for mobile devices in the world, as it has a large market share, both in Mexico and around the world. Firebase "Authentication" allows to keep track of all users, while Firebase "Realtime Database" is a low latency solution for mobile applications.

OBJECTIVES

The main objective of the application is to offer a technological tool that is used to create and find matches near your location, facilitating the occurrence of such "challenges" and promoting the meeting of users in public parks for sports.

GOALS

The main goal that the project seeks is to promote the culture of sports in young people, offering a platform to visualize the places available to play soccer in their city and giving the ability to schedule matches in those places, as well as consult the matches created by others users.

IMPACT OR BENEFIT ON THE SOLUTION TO A PROBLEM RELATED TO THE PRODUCTIVE SECTOR OR THE GENERATION OF TECHNOLOGICAL KNOWLEDGE

The successful implementation of this project would improve the sports culture of young people, while preserving that beautiful tradition of meeting in a park with friends to play soccer in their free time.

DEVELOPMENT METHODOLOGY

The development of this project required a good organization to be able to comply in a timely manner with its characteristics raised throughout the development, in addition to being attentive to the changes that may arise or the functionalities that will be implemented. An important part of the development was to keep backup copies of the work carried out, in order to continue the development regardless of the situation and be prepared for the risks that may arise.

SCHEDULE OF ACTIVITIES AND SCHEDULING

Gantt diagram

Act.	Duration (Days)	September			October		November		December
		11-17	18-24	25-28	1- 17	18-31	1-14	15-30	1-12
1	14								
2	14								
3	15								
4	13								
5	20								
6	7								

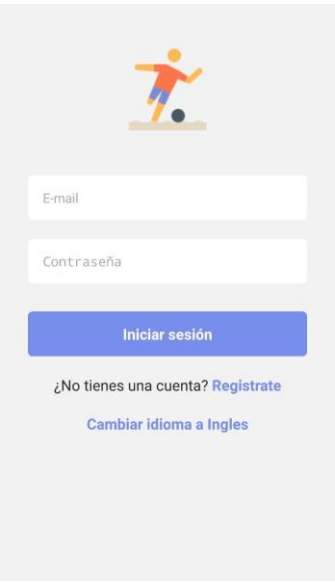
Activities realized

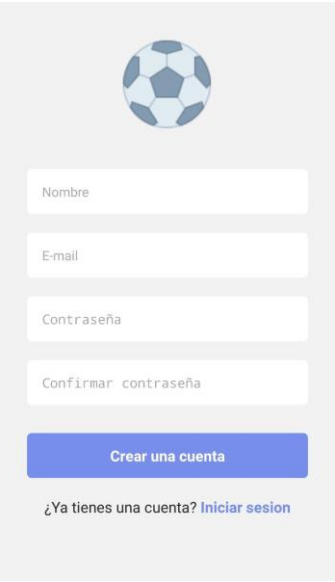
1. Ideas, Analysis and Planning
2. Design and Mockups
3. Development (Back-end)
4. Development (Front-end)
5. Tests, Q&A and Documentation
6. Deployment

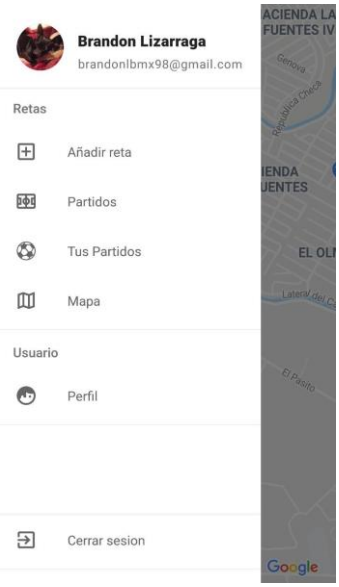
PRODUCTS

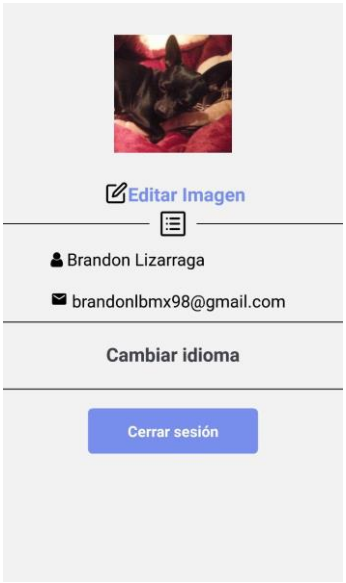
Phase I. User Interaction

In the first phase, the graphical interface was created that allows the user to move through the different sections of the application, as well as a registry where they can create a user or log in if they already have one.



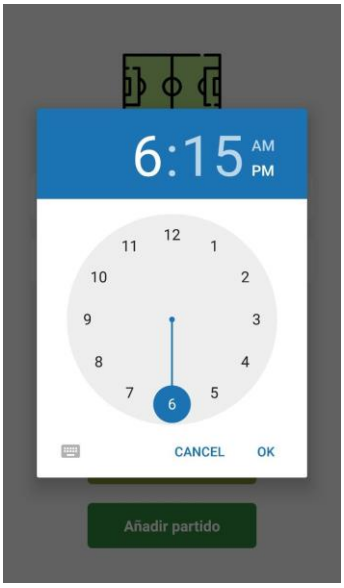


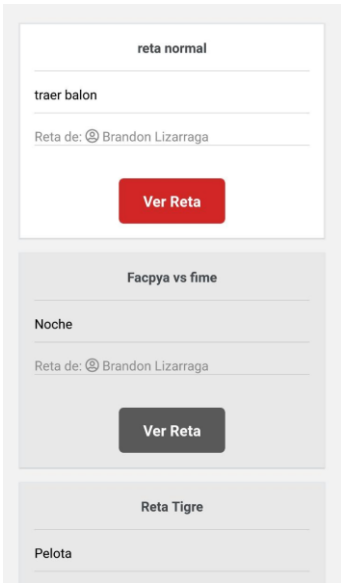




Phase II. Backend Interaction

During the second phase, the user can interact with other users, scheduling football matches where the description of this is specified and a schedule is determined, as well as viewing the matches scheduled by other users.





Database design

The database follows the following format, as shown in a JSON structure.

It is divided into 2 main components, the "Partidos" and "Users" component. Their names are indicative of their functions. "Matches" saves the matches that the user sends to the database and where the data is collected from it. While "Users" saves the data of user information and joined parties.

```
{
  "partidos" : {…
  },
  "users" : {…
}
```

In matches, matches are saved with a unique identifier that firebase provides.

```
"partidos" : {
    "-MN5KQd8hKxfFd11E2Vn" : {…
    },
    "-MN5MJu-YGbe8_NmfV3_" : {…
    },
    "-MNerO7_okDtpA_cZcbj" : {…
    }
  }
```

In a match, the data provided in the add match tab of the application is saved.

```
"-MN5KQd8hKxfFd11E2Vn" : {
      "anio" : 2020,
      "descripcion" : "Noche",
      "dia" : 28,
      "hora" : 18,
      "location" : {…
},
      "mes" : 10,
      "minutos" : 15,
      "postId" : "-MN5KQd8hKxfFd11E2Vn",
      "titulo" : "Facpya vs fime",
      "usuario" : "1cmggPFUd8TwPrGWkuGkTjVUOJy1",
      "usuarioDisplayName" : "Brandon Lizarraga",
      "usuarios" : [ "1cmggPFUd8TwPrGWkuGkTjVUOJy1" ],
      "usuariosDisplayName" : [ "Brandon Lizarraga" ]
    }
```

Under "users", user information such as image profile and matches created and joined are provided.


```
"1cmggPFUd8TwPrGWkuGkTjVUOJy1" : {
      "partidos" : {
        "-MN5KQd8hKxfFd11E2Vn" : {…
        },
        "-MNerO7_okDtpA_cZcbj" : {…
        }
      },
      "profile_image" : {
        "profile_image" : "https://firebasestorage.googleapis.com/v0/b/retasclub-
d5adc.appspot.com/o/e709c3d3-b716-4dc0-addc-068a45dd509b?alt=media&token=75c045b9-0b27-495c-
b660-0e83a68c5f2b"
      },
      "unidos" : {…
        }
```

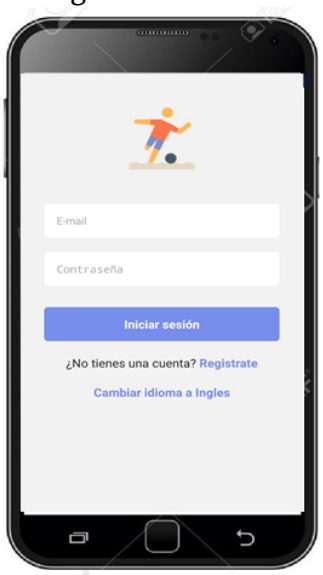
TECHNOLOGIES

Tipo	Software	Versión	Uso
Development environment	Visual Studio Code	1.51	Environment that allows the development of Application.
Editing tool	Office	360 (2016)	Completion of the necessary documentation for the Application.
API	Google Maps Google Places	NA	Viewing the map within the application, Viewing nearby parks with Google places.
Database	Firebase	16	Storage of information and users.
Programming language	Javascript	14.15.1 NodeJS	Programming of the app environment.
Framework	React Native and Expo	Expo 39.0 SDK 0.63 React Native	Working environment for Javascript and application export.

Screens

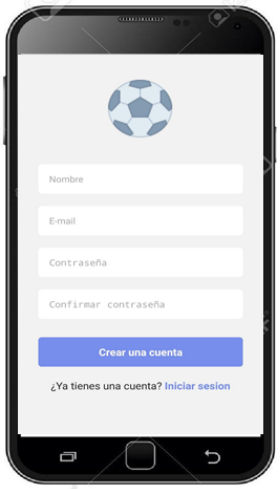
Main Screen

Here it is necessary to enter the username (or, failing that, the registered email) and the password. If you do not have an account, you must register.



Register Screen

The corresponding data is filled in and with that the user is registered.

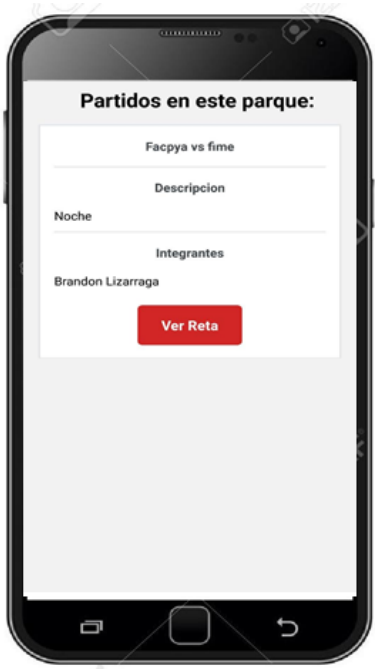


Main Screen

Once you enter your username and password, this screen is displayed, where you can see the games created by other users on a map.

Viewing matches

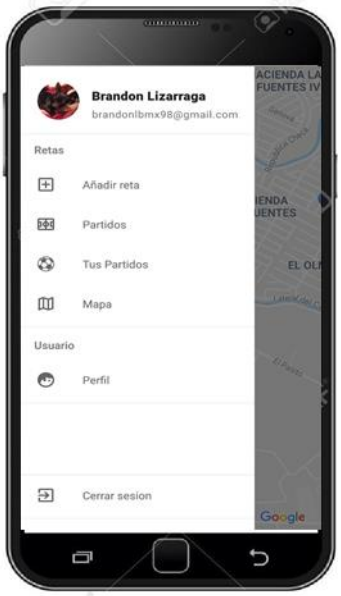
When selecting a game, this screen is shown where the games that will take place in that park are displayed.



Here you can view the information of a specific match and it gives you the option to join in case you are not the creator of it.

Menu

The menu section shows different options that the user can select.

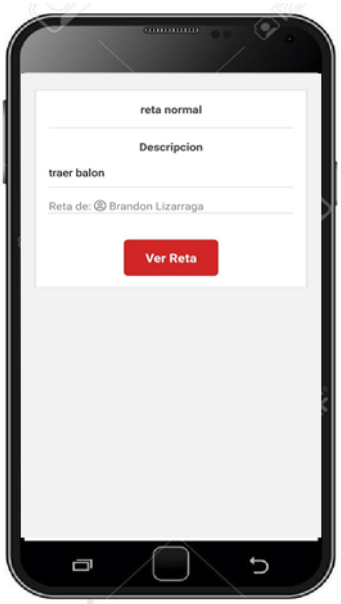


Profile screen

In this section the user information is displayed, here you can change the profile photo, change the language and log out.

Matches

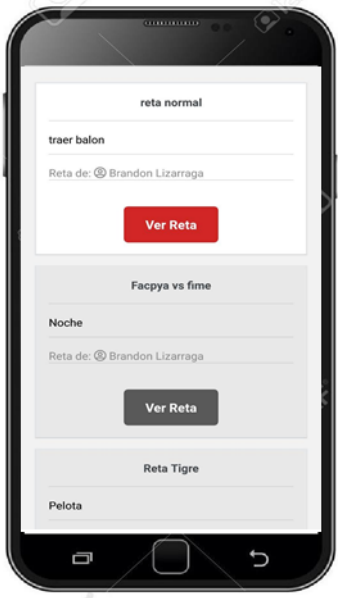
This tab shows all the matches that are about to take place, it should be noted that the matches of past dates and times are no longer shown.



Pressing a match shows its information.

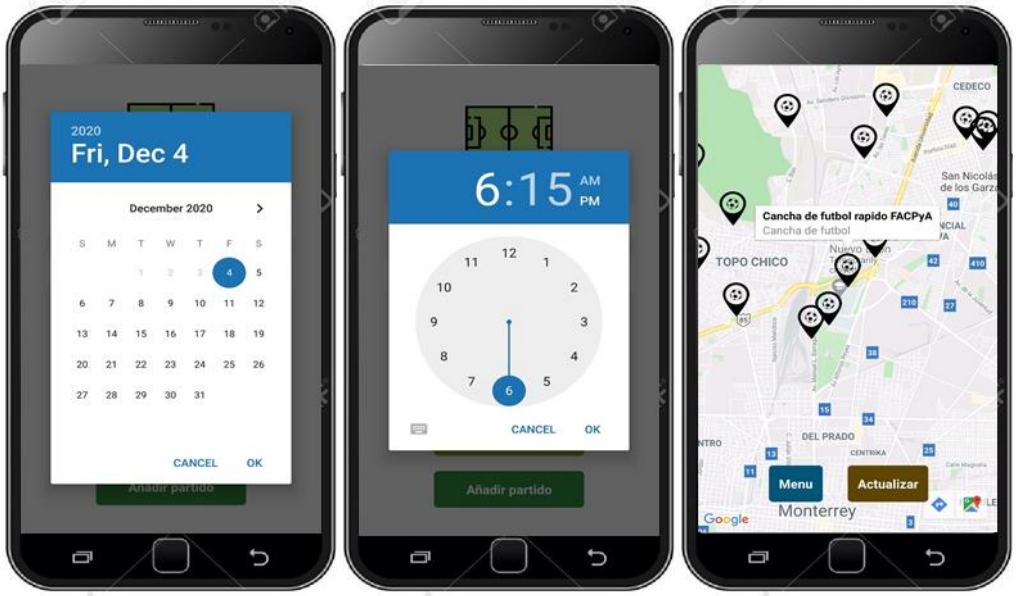
Your matches

In this tab you can see the matches created by you and in which you have joined. Matches created by you can be deleted before their date and past matches are displayed in gray.



Add match

In this tab you can schedule a match, selecting the date, time and place, as well as giving a general description of it.



USER MANUAL

OBJECTIVES

The main objective of the application is to offer a technological tool that is used to create and find matches near your location, facilitating the occurrence of such "challenges" and promoting the meeting of users in public parks for sports.

GOALS

The main goal that the project seeks is to promote the culture of sports in young people, offering a platform to visualize the places available to play soccer in their city and giving the ability to schedule matches in those places, as well as consult the matches created by other users.

ABSTRACT

The application has 4 main tabs:

The **Add challenge** screen contains a list where the user enters various values to add a match such as:

- Match Title
- Match Description
- Date
- Time
- Location

And at the end it contains an add match button.

The **Matches tab** contains a list of all current matches and a button to watch the match.

The **Your matches** tab contains matches that you have created and joined, as well as past-date matches that you have joined and created in.

The **Map screen** contains a map with all the created matches that are on date to join. Start with your location to find nearby matches.

Lastly, the **Profile** screen contains your profile information.

Requirements

The application was developed in React Native and Expo SDK. It can be used on any mobile device that has the Android operating system.

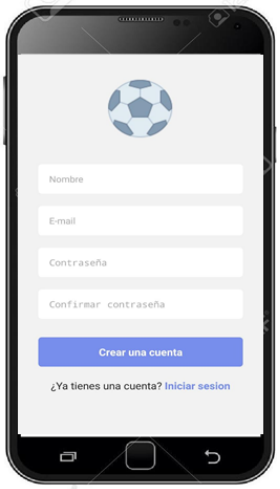
Screens

Main Screen

Here it is necessary to enter the username (or, failing that, the registered email) and the password. If you do not have an account, you must register.

Register Screen

The corresponding data is filled in and with that the user is registered.

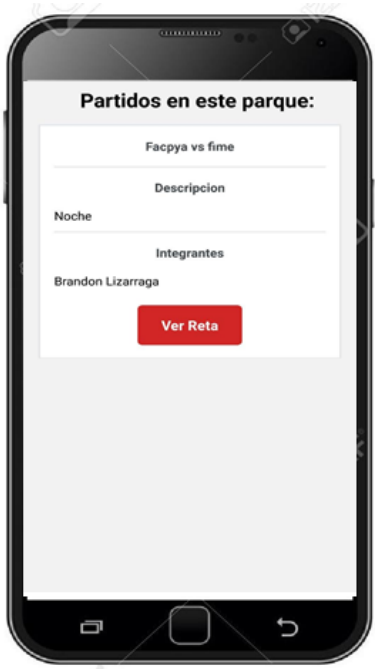


Main Screen

Once you enter your username and password, this screen is displayed, where you can see the games created by other users on a map.

Viewing matches

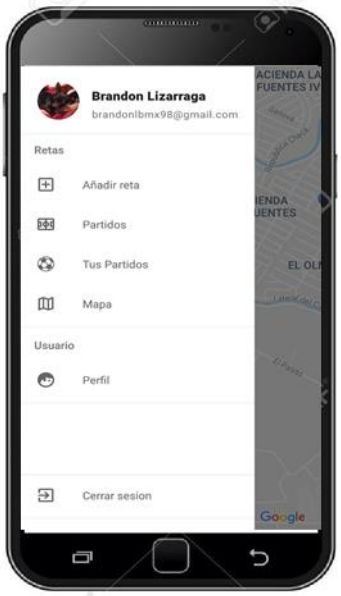
When selecting a game, this screen is shown where the games that will take place in that park are displayed.



Here you can view the information of a specific match and it gives you the option to join in case you are not the creator of it.

Menu

The menu section shows different options that the user can select.

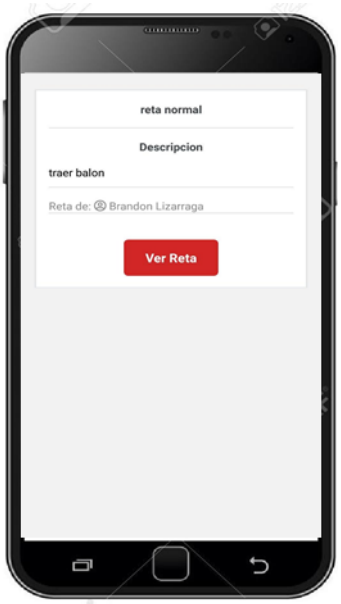


Profile screen

In this section the user information is displayed, here you can change the profile photo, change the language and log out.

Matches

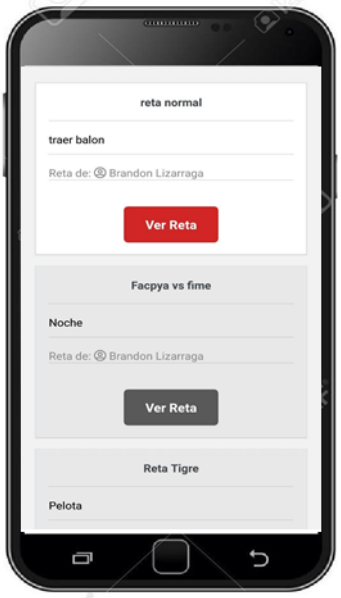
This tab shows all the matches that are about to take place, it should be noted that the matches of past dates and times are no longer shown.



Pressing a match shows its information.

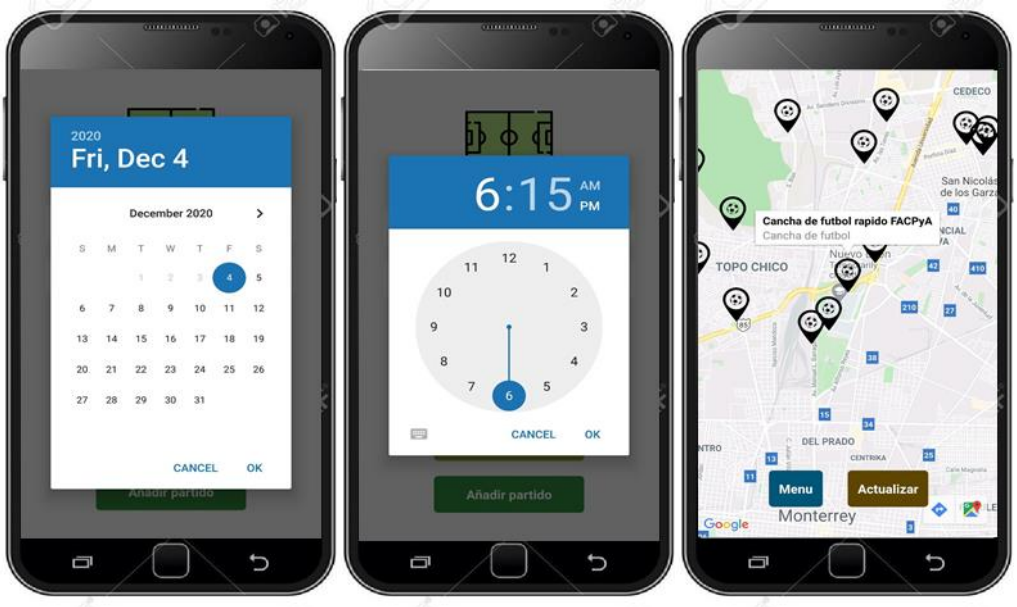
Your matches

In this tab you can see the matches created by you and in which you have joined. Matches created by you can be deleted before their date and past matches are displayed in gray.



Add match

In this tab you can schedule a match, selecting the date, time and place, as well as giving a general description of it.



Conclusions

The application manages to facilitate the scheduling of soccer matches, this because it allows users to view available places near their location, as well as matches scheduled on nearby dates. Thanks to the application, it is possible to promote the culture of sports and support so that the tradition of soccer challenges is not lost over time.

References

- Occhino, T. (2015). React Native: Bringing modern web techniques to mobile - Facebook Engineering. Retrieved 8 December 2020, from https://engineering.fb.com/2015/03/26/android/react-native-bringing-modern-web-techniques-to-mobile/
- API Reference - Expo Documentation. (2020). Retrieved 8 December 2020, from https://docs.expo.io/versions/latest/

- Introduction · React Native. (2020). Retrieved 8 December 2020, from https://reactnative.dev/docs/getting-started
- Google Cloud Platform. (2020). Retrieved 8 December 2020, from https://console.developers.google.com/

TECHNICIAN MANUAL

OBJECTIVES

The main objective of the application is to offer a technological tool that is used to create and find matches near your location, facilitating the occurrence of such "challenges" and promoting the meeting of users in public parks for sports.

GOALS

The main goal that the project seeks is to promote the culture of sports in young people, offering a platform to visualize the places available to play soccer in their city and giving the ability to schedule matches in those places, as well as consult the matches created by other users.

ABSTRACT

The application has 4 main tabs:

The **Add challenge** screen contains a list where the user enters various values to add a match such as:

- Match Title
- Match Description
- Date
- Time
- Location

And at the end it contains an add match button.

The **Matches tab** contains a list of all current matches and a button to watch the match.

The **Your matches** tab contains matches that you have created and joined, as well as past-date matches that you have joined and created in.

The **Map screen** contains a map with all the created matches that are on date to join. Start with your location to find nearby matches.

Lastly, the **Profile** screen contains your profile information.

TECHNOLOGIES

Tipo	Software	Versión	Uso
Development environment	Visual Studio Code	1.51	Environment that allows the development of Application.
Editing tool	Office	360 (2016)	Completion of the necessary documentation for the Application.
API	Google Maps Google Places	NA	Viewing the map within the application, Viewing nearby parks with Google places.
Database	Firebase	16	Storage of information and users.
Programming language	Javascript	14.15.1 NodeJS	Programming of the app environment.
Framework	React Native and Expo	Expo 39.0 SDK 0.63 React Native	Working environment for Javascript and application export.

Database design

The database follows the following format, as shown in a JSON structure.

It is divided into 2 main components, the "Partidos" and "Users" component. Their names are indicative of their functions. "Matches" saves the matches that the user sends to the database and where the data is collected from it. While "Users" saves the data of user information and joined parties.

```
{
  "partidos" : {…
  },
  "users" : {…
}
```

In matches, matches are saved with a unique identifier that firebase provides.

```
"partidos" : {
    "-MN5KQd8hKxfFd11E2Vn" : {…
    },
    "-MN5MJu-YGbe8_NmfV3_" : {…
    },
```

```
    "-MNerO7_okDtpA_cZcbj" : {…
    }
  }
```

In a match, the data provided in the add match tab of the application is saved.

```
"-MN5KQd8hKxfFd11E2Vn" : {
      "anio" : 2020,
      "descripcion" : "Noche",
      "dia" : 28,
      "hora" : 18,
      "location" : {…
},
      "mes" : 10,
      "minutos" : 15,
      "postId" : "-MN5KQd8hKxfFd11E2Vn",
      "titulo" : "Facpya vs fime",
      "usuario" : "1cmggPFUd8TwPrGWkuGkTjVUOJy1",
      "usuarioDisplayName" : "Brandon Lizarraga",
      "usuarios" : [ "1cmggPFUd8TwPrGWkuGkTjVUOJy1" ],
      "usuariosDisplayName" : [ "Brandon Lizarraga" ]
    }
```

Under "users", user information such as image profile and matches created and joined are provided.

```
"1cmggPFUd8TwPrGWkuGkTjVUOJy1" : {
      "partidos" : {
        "-MN5KQd8hKxfFd11E2Vn" : {…
        },
        "-MNerO7_okDtpA_cZcbj" : {…
        }
      },
      "profile_image" : {
        "profile_image" : "https://firebasestorage.googleapis.com/v0/b/retasclub-
d5adc.appspot.com/o/e709c3d3-b716-4dc0-addc-068a45dd509b?alt=media&token=75c045b9-0b27-495c-
b660-0e83a68c5f2b"
      },
      "unidos" : {…
        }
```

Application code

Among the most prominent elements of the code are:

App.jsx

```
export default function App() {

  const [loading, setLoading] = useState(true)
  const [user, setUser] = useState(null)

  return (
    <NavigationContainer>
      <Stack.Navigator screenOptions={{headerShown: false,}}>
        { user ? (
          <Stack.Screen name="Home">
            {props => <HomeScreen {...props} extraData={user} />}
          </Stack.Screen>
        ) : (
          <>
            <Stack.Screen name="Login" component={LoginScreen} />
            <Stack.Screen name="Registro" component={RegistrationScreen} />
            <Stack.Screen name="Home" component={HomeScreen} />
          </>
        )}
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Where the code is initialized with the App function and indicates the different screens of the application. The first screen where the user enters is the Login screen.

Login.jsx

```
export default function LoginScreen({ navigation }) {
    const [location, setLocation] = useState(null);
    const [language, setlanguage] = useState('es');

    async function getData(){
        try {
            const value = await AsyncStorage.getItem('@languageSet')
            if (value == null) {
                console.log("no language set, setting one")
                await AsyncStorage.setItem('@languageSet', 'es')
                const value = await AsyncStorage.getItem('@languageSet')
                setlanguage(value)
```

```
                changeLanguagePreset()
                return
            } else {
                setlanguage(value)
                changeLanguagePreset(value)
            }
        } catch (e) {
            console.log(e)
        }
    }

    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');


    (async () => {
        let { status } = await Location.requestPermissionsAsync();
        if (status !== 'granted') {
            setErrorMsg('Se necesita acceder a los permisos de ubicacion.');
        }

        let locationGet = await Location.getCurrentPositionAsync({});
        setLocation(locationGet);
    })();

    const [LANGcambiaridiomatexto, setLANGcambiaridiomatexto] = useState('Cambiar idioma a Ingles'
);
    const [LANGpasswordfield, setLANGpasswordfield] = useState('Contraseña');
    const [LANGIniciarses, setLANGIniciarses] = useState('Iniciar sesión');
    const [LANGNotienesunacuenta, setLANGNotienesunacuenta] = useState('¿No tienes una cuenta?');
    const [LANGregistrate, setLANGregistrate] = useState('Registrate');
    const [LANGUbicacion, setLANGUbicacion] = useState("Necesita tener ubicacion encendida para es
ta aplicacion.");
    const [LANGErrorVerif, setLANGErrorVerif] = useState("Error de verificacion");
    const [LANGVerifyEmail, setLANGVerifyEmail] = useState("Favor de verificar su cuenta por medio
 de su correo electronico.");



    function changeToSpanishLogin() {
        setLANGcambiaridiomatexto("Cambiar idioma a Ingles")
        setLANGpasswordfield("Contraseña")
        setLANGIniciarses("Iniciar sesión")
        setLANGNotienesunacuenta('¿No tienes una cuenta?')
        setLANGregistrate('Registrate')
        setLANGUbicacion("Necesita tener ubicacion encendida para esta aplicacion.")
        setLANGErrorVerif("Error de verificacion")
```

```
        setLANGVerifyEmail("Favor de verificar su cuenta por medio de su correo electronico.")
    }

    function changeToEnglishLogin() {
        setLANGcambiaridiomatexto("Change language to spanish")
        setLANGpasswordfield("Password")
        setLANGIniciarses("Log In")
        setLANGNotienesunacuenta("Don't have an account yet?")
        setLANGregistrate('Sign Up')
        setLANGUbicacion("Please turn on location services before logging in")
        setLANGErrorVerif("Verification Error")
        setLANGVerifyEmail("Please verify E-mail")
    }
    const onFooterLinkPress = () => {
        navigation.navigate('Registro', { language: language })
    }

    const onLoginPress = () => {
        if (location == null) {
            Alert.alert(LANGUbicacion)
            return
        }
        firebase
            .auth()
            .signInWithEmailAndPassword(email, password)
            .then(() => {
                // Listen for authentication state to change.
                firebase.auth().onAuthStateChanged((user) => {
                    if (user != null) {
                        console.log("We are authenticated now!");
                        let emailVerified = user.emailVerified;
                        if (emailVerified === true) {
                            navigation.navigate('Home', { user: user, location: location, language
: language })
                        }
                        else {
                            Alert.alert(
                                LANGErrorVerif,
                                LANGVerifyEmail,
                                [
                                    { text: 'OK', onPress: () => console.log('OK Pressed') },
                                ]
                            )
                            return
                        }
                    }
```

```
                });
            })
            .catch(error => {
                alert(error)
            })
    }

    function changeLanguagePreset(presetlanguage) {
        if (presetlanguage == 'es') {
            changeToSpanishLogin()
            return
        }
        if (presetlanguage == 'en') {
            changeToEnglishLogin()
        }
    }

    async function changeLanguage() {
        if (language == 'es') {
            setlanguage('en');
            changeToEnglishLogin()
            await AsyncStorage.setItem('@languageSet', 'en')
            Updates.reloadAsync()
            return
        }
        if (language == 'en') {
            setlanguage('es');
            changeToSpanishLogin()
            await AsyncStorage.setItem('@languageSet', 'es')
            Updates.reloadAsync()
        }
    }

    useEffect(() => {
        getData()
    }, []);
    return (
}
```

The user is presented with a Login form where they enter their data before entering the application. In addition to changing the language of the application and with a link to the registration screen.

RegistrationScreen.JSX

```
export default function RegistrationScreen({route, navigation }) {
    const language = route.params.language
    const [fullName, setFullName] = useState('')
    const [email, setEmail] = useState('')
    const [password, setPassword] = useState('')
    const [confirmPassword, setConfirmPassword] = useState('')

    const [LANGpasswordfield, setLANGpasswordfield] = useState('Contraseña');
    const [LANGNombreField, setLANGNombreField] = useState('Nombre');
    const [LANGConfirmarC, setLANGConfirmarC] = useState('Confirmar contraseña');
    const [LANGCrearCuenta, setLANGCrearCuenta] = useState('Crear una cuenta');
    const [LANGYatienesunaC, setLANGYatienesunaC] = useState('¿Ya tienes una cuenta?');
    const [LANGiniciarses, setLANGiniciarses] = useState('Iniciar sesion');
    const [LANGContranocoinciden, setLANGContranocoinciden] = useState('Las contraseñas no coincid
en.');
    const [LANGCorreoVerif, setLANGCorreoVerif] = useState('Se ha enviado un correo de verificació
n a tu dirección de correo electrónico. Por favor, revisa to correo para terminar de crear tu cuen
ta');

    function changeToSpanishLogin() {
        setLANGpasswordfield("Contraseña")
        setLANGNombreField("Nombre")
        setLANGConfirmarC("Confirmar contraseña")
        setLANGCrearCuenta("Crear una cuenta")
        setLANGYatienesunaC("¿Ya tienes una cuenta?")
        setLANGiniciarses("Iniciar sesion")
        setLANGContranocoinciden('Las contraseñas no coinciden.')
        setLANGCorreoVerif("Se ha enviado un correo de verificación a tu dirección de correo elect
rónico. Por favor, revisa to correo para terminar de crear tu cuenta")
    }

    function changeToEnglishLogin() {
        setLANGpasswordfield("Password")
        setLANGNombreField("Name")
        setLANGConfirmarC("Confirm Password")
        setLANGCrearCuenta("Create an account")
        setLANGYatienesunaC("Already have an account?")
        setLANGiniciarses("Log in")
        setLANGContranocoinciden("Passwords don't match. Try again.")
        setLANGCorreoVerif("An E-mail has been sent. Please check to access your account.")
    }


    function changeLanguagePreset(){
```

```
        if (language == 'es') {
            changeToSpanishLogin()
            return
        }
        if (language == 'en') {
            changeToEnglishLogin()
        }
    }

    const onFooterLinkPress = () => {
        navigation.navigate('Login')
    }

    const VerificacionToast = () => {
        if (Platform.OS === 'android') {
            ToastAndroid.show(LANGCorreoVerif, ToastAndroid.LONG);
        }
        var user = firebase.auth().currentUser;

        user.sendEmailVerification().then(function () {
            // Email sent.
        }).catch(function (error) {
            // An error happened.
        });
    };

    const onRegisterPress = () => {
        if (password !== confirmPassword) {
            alert(LANGContranocoinciden)
            return
        }
        firebase
            .auth()
            .createUserWithEmailAndPassword(email, password)
            .then(() => {
                firebase
                    .auth()
                    .signInWithEmailAndPassword(email, password)
                var user = firebase.auth().currentUser;
                user.updateProfile({
                    displayName: fullName
                }).then(function () {
                    // Update successful.
                }).catch(function (error) {
                    // An error happened.
                });
```

```
                var profile_image = firebase.database().ref("users/" + user.uid + "/" + "profile_i
mage")

                profile_image.set({
                    profile_image: "https://firebasestorage.googleapis.com/v0/b/retasclub-
d5adc.appspot.com/o/soccer-player.png?alt=media&token=4cf53cc3-cc8c-44c3-83dc-c2278a2d472e",
                })
                firebase.auth().signOut()
                navigation.navigate('Login')
                VerificacionToast()
            })
            .catch((error) => {
                alert(error)
            });
    }

    useEffect(() => {
        changeLanguagePreset()
    }, []);

    return (
        <View style={styles.container}>
            <KeyboardAwareScrollView
        </View>
    )
}
```

HomeScreen.jsx

```
return (
       <Drawer.Navigator initialRouteName="MapaMain" drawerType="slide" drawerContent={props => <DrawerContent {...props} routes={object} /
>}>
           <Drawer.Screen name="Añadir reta" component={AddPartidaScreen} initialParams={{ user, language: languageObject }} />
           <Drawer.Screen name="ParqueView" component={ParqueView} initialParams={{ user: user, location: location, language: languageObjec
t }} />
           <Drawer.Screen name="Mapa" component={MapaScreen} initialParams={{ user: user, location: location, language: languageObject }} /
>
           <Drawer.Screen name="MapaMain" component={MapaMain} initialParams={{ user: user, location: location, language: languageObject }}
 />
           <Drawer.Screen name="Perfil" component={PerfilScreen} initialParams={{ user, language: languageObject }} />
           <Drawer.Screen name="Partidos" component={PartidosScreen} initialParams={{ user, language: languageObject }} />
           <Drawer.Screen name="TusPartidos" component={TusPartidos} initialParams={{ user, language: languageObject }} />
           <Drawer.Screen name="TusPartidosView" component={TusPartidosView} initialParams={{ language: languageObject }} />
           <Drawer.Screen name="Reta" component={retaScreen} initialParams={{ user, language: languageObject }} />
           <Drawer.Screen name="RetaDone" component={retaScreenDone} initialParams={{ user, language: languageObject }} />
       </Drawer.Navigator>
   );
```

The different components of the main application that starts when the user gives to log in.

- AddPartidaScreen: Screen where a form is given to add matches to the database and the map.
- ParqueView: Screen where the parks are seen when one clicks on the map.
- MapaScreen: Main screen where the map is shown with the games displayed on it.
- ProfileScreen: Screen where the user's profile is displayed.
- PartidosScreen: Screen where matches are shown without the map.
- TusPartidos: Screen where matches that you have created or joined are shown.
- TusPartidosView: Screen where the individual match that is clicked on the TusPartidos screen is shown.
- retaScreen: Screen where the individual game is shown in general to the public, there is a join button.
- retaScreenDone: Screen showing matches that have passed the date and in which they cannot be joined.

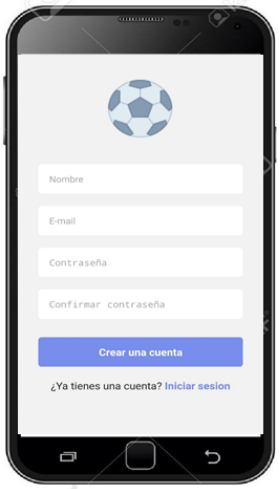
Screens

Main Screen

Here it is necessary to enter the username (or, failing that, the registered email) and the password. If you do not have an account, you must register.

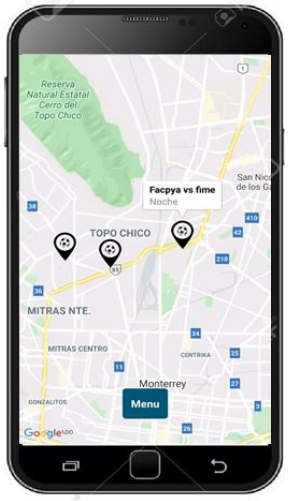
Register Screen

The corresponding data is filled in and with that the user is registered.



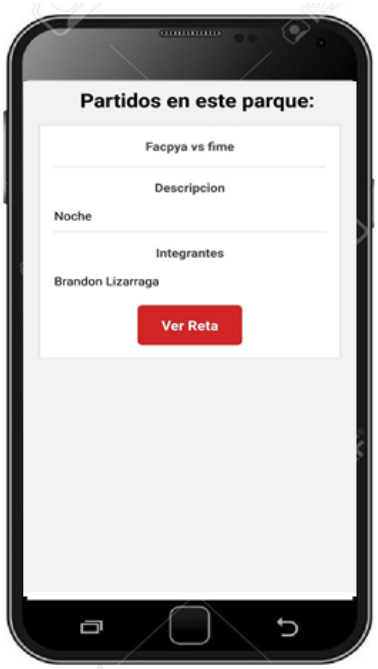
Main Screen

Once you enter your username and password, this screen is displayed, where you can see the games created by other users on a map.



Viewing matches

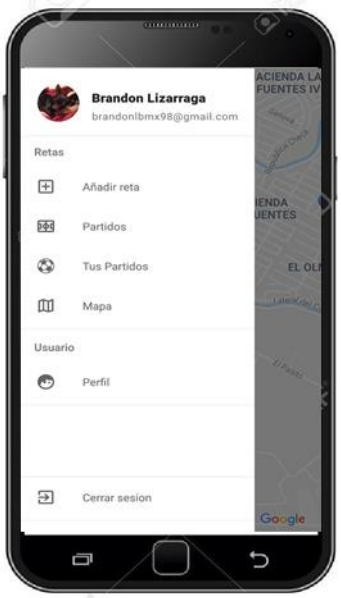
When selecting a game, this screen is shown where the games that will take place in that park are displayed.



Here you can view the information of a specific match and it gives you the option to join in case you are not the creator of it.

Menu

The menu section shows different options that the user can select.

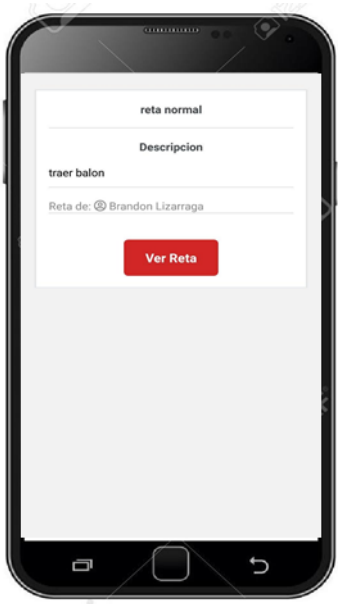


Profile screen

In this section the user information is displayed, here you can change the profile photo, change the language and log out.

Matches

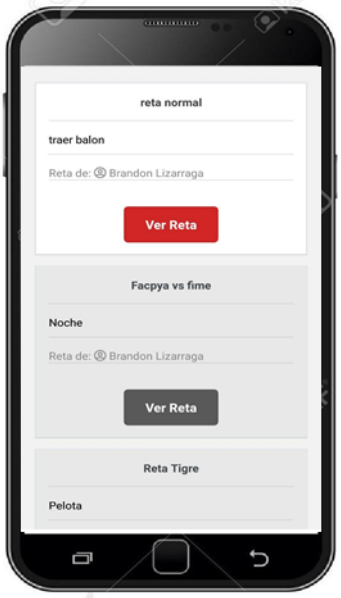
This tab shows all the matches that are about to take place, it should be noted that the matches of past dates and times are no longer shown.



Pressing a match shows its information.

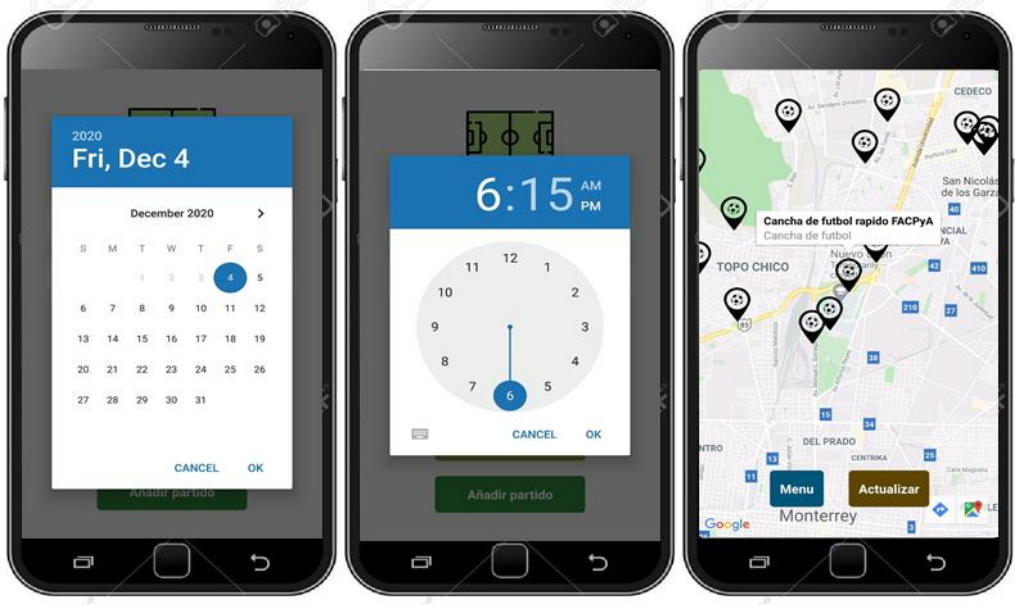
Your matches

In this tab you can see the matches created by you and in which you have joined. Matches created by you can be deleted before their date and past matches are displayed in gray.



Add match

In this tab you can schedule a match, selecting the date, time and place, as well as giving a general description of it.



Conclusions

The application manages to facilitate the scheduling of soccer matches, this because it allows users to view available places near their location, as well as matches scheduled on nearby dates. Thanks to the application, it is possible to promote the culture of sports and support so that the tradition of soccer challenges is not lost over time.

References

- Occhino, T. (2015). React Native: Bringing modern web techniques to mobile - Facebook Engineering. Retrieved 8 December 2020, from https://engineering.fb.com/2015/03/26/android/react-native-bringing-modern-web-techniques-to-mobile/
- API Reference - Expo Documentation. (2020). Retrieved 8 December 2020, from https://docs.expo.io/versions/latest/
- Introduction · React Native. (2020). Retrieved 8 December 2020, from https://reactnative.dev/docs/getting-started
- Google Cloud Platform. (2020). Retrieved 8 December 2020, from https://console.developers.google.com/