TEC | Tecnológico de Costa Rica

Área académica de ingeniería en computadores

Bases de Datos (CE3101)

Título: Stravia TEC

Estudiantes:

Jonathan García Ugalde

Ronny Aguilar Barahona

Roger Mora González

Profesor:

Marco Rivera Meneses

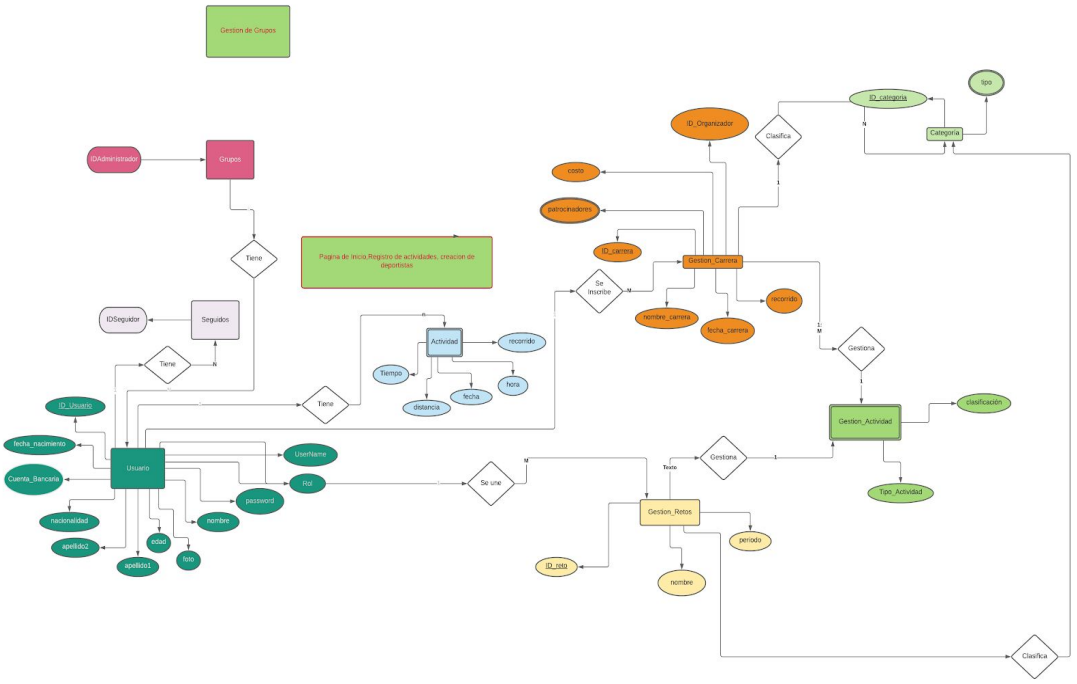
Periodo lectivo:

II Semestre 2020

Grupo:

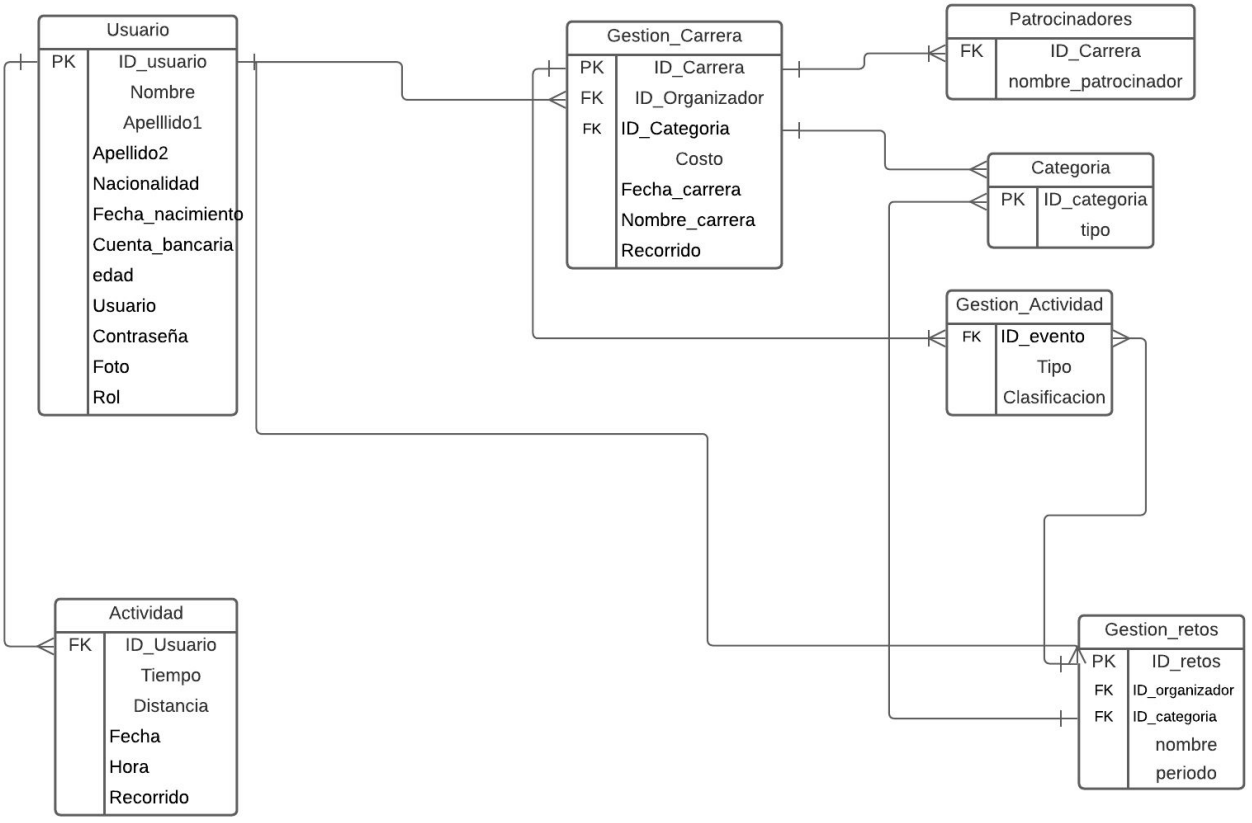
02

Modelo Conceptual



Para un mejor acceso al diagrama Conceptual visite :https://lucid.app/lucidchart/invitations/accept/8d9489a7-7eac-4855-9129-a8d3ae6dddd1

Modelo Relacional.



Para una mejor visualización de diagrama relacional, visite:https://lucid.app/lucidchart/invitations/accept/9c797f75-2f86-460d-8b86-49843b7cc50d

Descripción de los métodos implementados.

Backend (Controller)

Para esta sección se va a documentar- la funcionalidad de los controladores que se encuentran desde la sección del API y cual es su labor funcional desde el punto funcional para poder comunicarse con el proyecto en angular

ActividadController: En esta clase se Gestiona las inserciones de las actividades de las personas, ya sea manualmente o las registradas por la aplicación móvil.

AmigosController: En esta Clase se manejan métodos para la inserción de amigos, el borrado de amigos y la búsqueda de amigos.

CategoríaController: En esta clase se manejan las categorías que se pueden asignar en las carreras y retos, una vez almacenados estos pueden asignarse a retos o carreras dependiendo del organizador.

ContactenosController: En esta clase se manejan los métodos post y get, los cuales se insertan datos de información de contacto y se envían a la aplicación web, respectivamente,

GestíonDeActividadController:en esta Clase se encarga tomar las actividades registradas por el usuario y proyectarlas en la página web.

GestionDeCarrerasController: en esta clase se usan los protocolos get, put,post, del, para que mediante sus respectivos métodos se puedan registrar, editar, buscar o eventualmente borrar una carrera.

GruposPrivadosPorCarreraController: En esta Clase se insertan los usuarios registrados en una carrera que tienen acceso al grupo de la carrera, mediante el

método **add(Gruposprivadosporcarrera grupo)**, el cual recibe por parámetros un objeto encapsulado de tipo grupos que contienen el id del usuario, id del grupo y el id de la carrera. también se puede eliminar el usuario con el método **del(int id)** que recibe por por parámetros el id del usuario que se desea eliminar. por otra parte también se pueden obtener los grupos de las carreras con el método **get(),** el cual devuelve todos los grupos que se pueden acceder por los diferentes carreras.

GruposPrivadosPorCarreraController: En esta Clase se insertan los usuarios registrados en un reto a que tienen acceso al grupo del reto, mediante el método **add(Gruposprivadosporreto grupo)**, el cual recibe por parámetros un objeto encapsulado de tipo grupos que contienen el id del usuario, id del grupo y el id del reto. también se puede eliminar el usuario con el método **del(int id)** que recibe por por parámetros el id del usuario que se desea eliminar. Por otra parte también se pueden obtener los grupos de los retos con el método **get(),** el cual devuelve todos los grupos que se pueden acceder por los diferentes retos.

IncripciónDeCarreraController: Esta Clase se encarga del proceso de tomar el identificador de un usuario y proceder a inscribirlo en una carrera, mediante un método que toma las un objeto de tipo **InscripDecarrera** el cual es el que recibe los datos desde el los post.Además se puede cancelar la inscripción de ese usuario, es decir que cancela su participación mediante el método **delete** HttpDelete("del/{id}")]

IncripciónDeRetosController: La clase inscribirse de retos trabaja de manera similar al de la clase **IncripciónDeCarreraController,** con la diferencia que el Objeto con el que se trabaja es el de la **InscripDeretos.**

PatrocinadoresController:

Esta Clase Cumple con dos funciones específicas, la primera poder ingresar patrocinadores a un reto específico con el método **[HttpPost] post(Patrocinadores)** para mostrarle a los diversos retos o carreras, los

patrocinadores de esta. es Utilizado para obtener todos los patrocinadores mediante el método get() con la ayuda del [HttpGet].

PatrocinadoresRetosController: Esta clase se encarga de realizar asignaciones de los patrocinadores, haciendo uso de los métodos **Gestionretos Gestionreto { get; set; }, Patrocinadores Patrocinador { get; set; }**, Estos

PatrocinadoresPorCarreraController: Esta Clase se encarga del proceso de asignación de patrocinadores a las carreras. Haciendo uso de un objeto de tipo **PatrocinadoresCarrera** donde se realizan asignaciones a los objetos de tipo GestionCarrera y patrocinador.

PatrocinasdoresPorRetoController: Esta Clase se encarga del proceso de asignación de patrocinadores a las carreras. Haciendo uso de un objeto de tipo **PatrocinadoresObjeto** donde se realizan asignaciones a los objetos de tipo GestionRetoy patrocinador.

Backend (Model)

Se creó un proyecto en Angular. La página se divide en varios componentes, donde cada componente corresponde a una vista o “sub-página”. Estos componentes tienen su propio directorio, un archivo css donde se ponen estilos únicos para esa vista, un archivo ts desde donde se consume la información desde el web service, y un web service encargado de consumir la información directamente desde el api en formato Json.

Además se tiene un módulo routing, encargado de trabajar el redireccionamiento interno de la página (urls y extensiones). Cada vez que se ingresa una extensión no reconocida, se devuelve a /home.

Descripción de las estructuras de datos desarrolladas.

Como es sabido los archivos gpx, se almacenan en formato de estructura xml, para el caso del app fue necesario su utilización, esto con el fin de poder realizar el mecanismo de almacenado en la aplicación móvil, la cual se encarga de posteriormente sincronizar con la base de datos de postgre.Este es un ejemplo:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<gpx xmlns="http://www.topografix.com/GPX/1/1" creator="byHand"
version="1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.topografix.com/GPX/1/1
http://www.topografix.com/GPX/1/1/gpx.xsd">

  <wpt lat="39.921055008" lon="3.054223107">
    <ele>12.863281</ele>
    <time>2005-05-16T11:49:06Z</time>
    <name>Cala Sant Vicenç - Mallorca</name>
    <sym>City</sym>
  </wpt>
```

```
</gpx>
```

Tablas: Haciendo uso de un motor de bases de datos llamado postgresql se procedió a diseñar una solución del proyecto basado en el modelo conceptual y relacional con el fin de definir las entidades pertinentes del proyecto. Dentro de ellas se almacenan los datos que se registren de la página web y posterior a ello se procede a realizar las consultas respectivas de inserción, actualización y borrado.

Descripción detallada de la arquitectura desarrollada.

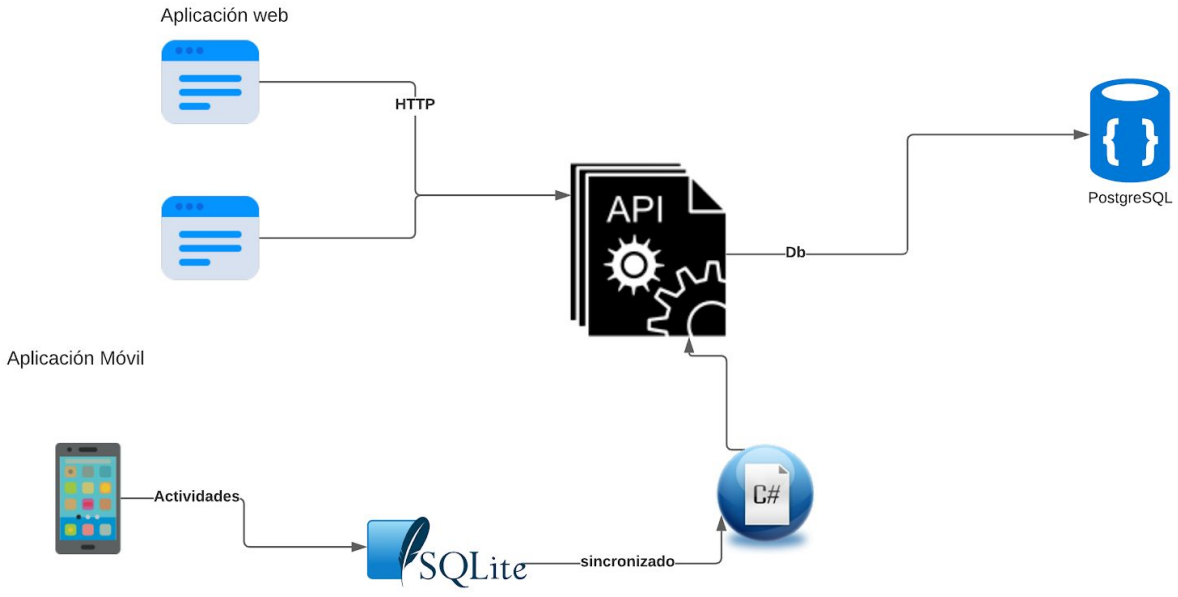
La arquitectura de este proyecto se compone de 4 módulos principales:

El primero es una **Aplicación web,** en esta sección se atribuye todo lo relacionado con el view (gestión el comportamiento visual del proyecto).Acá se brinda la una experiencia de comunicación entre el usuario Final y la página Web, se busca que en este modulo lasexperiencais con el usuario sean lo más intuitivas y con el amigables, para que su interacción no sea entorpecida.
Relacionado con las conexiones: en esta sección es donde se conectan todas las vistas del proyecto para que se pueda realizar un intercambio de datos con la base de datos mediante el protocolo HTTP, de esta manera se realizan las diferentes consultas en la base de datos,eso sí siempre gestionada por el API.

el Siguiente Módulo es el del **Rest API**(*application programming interfaces*),En este caso esta va a ser considerada por el contrato entre un proveedor de información, es decir la base de datos y los usuarios que para este caso va ser la aplicación web, aquí se va a establecer el contenido que se requiere del consumidor del servicio y el app.
En otras palabras, Esta va ser el responsable de con una computadora o un sistema para obtener datos o ejecutar una función, el API es quien va a permitir la función de traductor de mensaje haciendo uso del protocolo HTTP(Hypertext Transfer Protocol)

El módulo de la **bases de datos** consta de un servidor en un motor de base de datos de postgreSQL, el cual contiene las tablas modeladas en el modelo relacional la cuales en u estructura se crearon con la finalidad de ser una herramienta que gestione de forma fácil la integración actualización y borrado de datos en la aplicación web, para acceder a la comunicación de esta base de datos con la página web y el app, es donde participa la el Gestor de operaciones Controller de API, aquí se consumen y se crean las consultas necesarias para el intercambio de información.

Por último Se encuentra el módulo de la **aplicación móvil** relacionado con la base de datos SQlite, en esta sección la funcionalidad del app consiste en registrar los recorridos de un usuario, este recorrido se interpreta mediante un API de google y Spotify la cual registra los recorridos, los almacena en una base de datos junto a un archivo gpx del recorrido y este lo actualiza en la página web mediante el Controller en el rest API.



Problemas conocidos:

Problemas del app móvil

después de haberse analizado la estructura de la posible sincronización de la con el grupo vía whatsapp con el grupo se aprobó la posibilidad de poder gestionarse en formato XML los datos desde la base de datos SQLite al API para que se sincronizarla con la base de datos en postgre, el problema surge por la poca compatibilidad que tenia la version de flutter, com.spotify.android.app remote.api fue una herramienta importante para la solución del problema.

Problemas encontrados:

https://github.com/rogmg03/StraviaTEC/issues/1

Documentación que evidencie el trabajo en equipo:

Definición de roles

Project Manager:

Encargado: Roger Mora González

Dentro de las principales funciones se encuentran:

- Será la línea directa de comunicación entre el profesor y el grupo de trabajo.
- Se encargará de establecer las tareas correspondientes a cada persona del grupo.
- Se encargará de velar por que las tareas asignadas se cumplan en el tiempo requerido.
- Cumplir con las tareas asignadas referentes a la programación de la aplicación.

Carga de archivos:

Encargado: Jonathan García Ugalde

Dentro de las principales funciones se encuentran:

- Empaquetar los archivos necesarios para que el sistema funcione de la mejor manera a la hora de la revisión.
- Detallar si antes de la ejecución del sistema será necesario instalar algún complemento adicional.
- Cumplir con las tareas asignadas referentes a la programación de la aplicación

Minutas de sesiones:

Encargado: Ronny Aguilar Barahona

Dentro de las principales funciones se encuentran:

- Resumir las actividades realizadas durante las reuniones.
- Resumir los problemas encontrados durante las reuniones.
- Llevar el control de horas invertidas en cada reunión.
- Cumplir con las tareas asignadas referentes a la programación de la aplicación

Plan de comunicación

Se estiman dos sesiones mínimas por semana en los días Martes y Jueves a las 7:00 pm, Se trabaja utilizando metodologías ágiles “Scrum”, por lo que estas sesiones se utilizarán para validar el avance de cada persona del grupo, para discutir inconvenientes presentes en las tareas asignadas y así identificar las posibles soluciones.

Como plan de contingencia para aquellas sesiones que no se puedan realizar por factores académicos o personales,primero se debe contemplar el impacto que este causa al proyecto en cuestión de tiempos de entrega y así validar si para reponer esa sesión basta con otra sesión en un dia a convenir o bien se deben hacer dos sesiones, una con el objetivo de replantear las tareas asignadas y las fechas de entrega y la otra como una sesión normal de entrega de avances.

Estas sesiones de contingencia se deben de programar en la misma semana.

Nota importante, el plan está sujeto a variaciones que se tomen de acuerdo a las reuniones semanales.

Plan de actividades

Distribución de responsabilidades.

Actividad	Estimación	Responsable
Creación del API y conexión con PostgreSQL	8h	Roger,Ronny
Modelos y controladores del API	1h por modelo	Roger
Creación base de datos	4h	Ronny
Relleno de datos	6h	Ronny
Proyecto inicial en angular	4h	Roger,
Template página de inicio	2h	Ronny
Template Log In/Sign in	2h	Ronny
Template del “Muro”	3h	Roger,Ronny
Routing entre las vistas	2h	
API consumers en Angular	4h	Roger,Ronny, Jonathan
Template buscador de usuarios	2h	Ronny
Template “Registrar actividad”	2h	Ronny
Template “Inscribirse a una carrera/reto”	2h	Roger
Template “Vista de	2h	Roger

grupos”		
Template “Buscador de carreras”	4h	Ronny
Template “Gestión de carrera”	4h	Ronny
Template “Aceptar inscripción”	2h	Roger
Asociar dinero (virtual, no real) con la plataforma	6h	Roger,Ronny,Jonathan
Template “Gestión de retos”	4h	Ronny
Investigar servicios de reportes	6h	Roger,Ronny
Implementar servicios de reportes	6h	Ronny
Aplicación móvil base	4h	jonathan
Template de “log in” para la aplicación	2h	jonathan
Generación de .gpx desde app móvil	6h	jonathan
Template de “registrar actividad” app móvil	4h	jonathan
app,web, Creación SQL Lite DB y sincronización con postgresql	8h	jonathan,Roger,Ronny
CRUD para app móvil	6h	jonathan

Asignación de tareas completitud de informe ejecutivo 2:

Actividad	Estimación	Porcentaje de avance (4/11/2020)	Actualizaciones	Porcentaje de avance (11/11/2020)
Creación del API y conexión con PostgreSQL	8h	40%	Se crean los consumer funcionales del usuario o deportista	60%
Creación base de datos	4h	100%	Completa	100%
Relleno de datos	6h	10%	Se crean varios usuarios de prueba utilizando postman para consumir los services	30%
Proyecto inicial en angular	4h	100%	Completado	100%
app,web, sincronizacion SQlite y PostgreSQL	8h	0%	No ha habido sincronización con api ni bases de datos	0%
CRUD para app móvil	6h	40%	Se guardan valores que insertan manualmente más no los generados por el recorrido	50%
Aplicación base, componentes dependencias, pruebas, botones básicos	4h	100%	Se migra la Base de Datos a Kotlin debido a problemas con el archivo gpx, funcionan con normalidad los CRUD, básicos,	100%

			pendiente depuración del almacenamiento de la ruta.	
Template de “log in” para la aplicación, versión visual	2h	45%	se guarda en la base de datos el usuario y contraseña, no se ha podido resolver la sección de recuperación de contraseña	55%
App móvil, (Kotlin), investigación sobre archivos gpx registros de rutas,	5h	10%	del dia 6 de noviembre al 10 se procede a migrar la estructura para una mejor implementación en Kotlin	40%
app móvil, almacenamiento de rutas en mapas por deportista	3h	10%	Se crea una clase que se encargue de realizar la obtención de la rutas en coordenadas y convertirlas en formato gpx	45%
app, registro usando el gps en flutter para modelo de stravia(archivo gpx)	4h	10%(meramente investigativo)	Desde la parte de la aplicación móvil no se ha hecho el merge para unir el punto anterior debido a que se encuentra en desarrollo	35%
Generación de .gpx desde app móvil	6h	0%	se estima que para el viernes 13/11/2020 se pueda generar desde el app móvil el archivo gpx de una ruta x	40%

			para ser almacena	
Template de “registrar actividad” app móvil	4h	0%	template inicial para ciclistas con los atributos del cliente	40%

Bitácora

Actividad	Fecha	Explicación	Encargado	Horas
Investigación de Conexion postgressql	27/10/2020	Se investiga cómo conectarse a la Base de datos utilizando node.js	Ronny	2
Creación del diagrama conceptual	27/10/2020	Creación del diagrama conceptual a utilizar para crear la base de datos	Ronny Roger Jonathan	1
Pruebas de conexion DB	28/10/2020	Se hacen pruebas de conexion a postgres utilizando repositorio pg y express	Ronny	3
Edición de plan de trabajo	29/10/2020	respecto al feedback aportado por el profesor se procede a depurar plan de trabajo.	Ronny, Roger, jonathan	
Investigación gpx en html	1/11/2020	Se investigó cómo cargar archivos GPX en	Roger	3h

		la página web.		
conexión de la app base a la base de datosSQlite	1/11/2020	Se procedió a crear el modelo de la base de datos desde el editor usando plugins para facilitar uso de SQlite	Jonathan	5h/ incluye solución de errores
App móvil, (flutter), investigación sobre archivos gpx registros de rutas.	5/11/2020	Se toma la decisión de migrar el diseño de la aplicación a Klotin, debido a errores de en xml de gpx	jonathan	3h
app móvil, almacenamiento de rutas en mapas por deportista	6/11/2020	se investiga la implementación del almacenamiento de las rutas y la exportación de las mismas a la base de datos	jonathan	3h
Diseño de vistas (login)	6/11/2020	Diseño web vista de login	Roger	2h
consultas de un posible método de sincronización de la base de datos SQlite con el API		después de analizar la estructura de los archivo gpx se consulta la idea de gestionar el intercambio de datos mediante un archivo xml para la sincronización	jonathan	3h
soluciòn de problema de XML del gpx	7/11/2020-10/11/2020		jonathan	6h, 2/cd

Diseño vistas (Contactanos y Amigos)	7/11/2020	Diseño de las vistas de contáctanos y amigos (seguidores y seguidos)	Roger	3h
Creacion Webservice usuarios	7/11/2020-10/11/2020	Se crean los api consumer de la base de datos encargados de extraer, crear, actualizar y eliminar datos referentes a los usuarios dentro de la base de datos	Ronny	8 h
Diseño de vistas (Muro, Carreras, Retos)	11/11/2020-12/11/2020	Se diseñó el muro o página principal, las vistas de inscripción de carreras, inscripción de retos y la visualización de ambas categorías.	Roger	8h
Problemas con el app gpx	12/11/2020-17/	no se logra poder guardar los datos del gps en el gpx, existe un atraso que deja congelado el avance se los requerimientos del app,	jonathan	7h
Diseño de vistas (Grupos)	15/11/2020	Se diseñó la vista de grupos	Roger	2h
Creación de controladores	19/11/2020	se procedió a diseñar y depurar controladores	Ronny, jonathan	5h

		faltantes para la comunicación con la aplicación web		
Implementación de ngFor para cargar la información en las plantillas	17/11/2020 - 20/11/2020	Se extrae la información utilizando los API consumers dentro del html con ngFor, ngIf, {{dato}}, etc. Presenta algunos errores con {{dato.parámetro}}	Roger, Ronny	8h
Documentación	21/11/2020	se depura y mejora la documentación Técnica	Jonathan	5h

Minutas

13/11/2020

la sesión se trató de dudas y consultas sobre la aplicación móvil y la explicación del apicore y los consumer del service en angular

17/11/2020

se coordina una posible solicitud de tiempo, debido a los imprevistos presentados en el proyecto.

19/11/2020

reunión donde se le solicita colaboración a los compañeros de trabajo para finalizar controladores, se detallan mejoras de la sección visual

Conclusiones y Recomendaciones del Proyecto.

Al finalizar el proceso de desarrollo del proyecto se infiere la necesidad de utilizar una herramienta de gestión de proyectos como gitKrakenBoard la cual ayudo al equipo a organizarse en lo posible. esto porque en nuestro caso ser un grupo conformado por tres miembros y no cuatro como se recomienda en el enunciado incluye un ⅓ de labor extra por mienbro del trabajo.

Se se hizo uso de visual Studio para el desarrollo del restServices lo cual permitió poder realizar este proyecto con una tecnología que se utiliza en muchas compañías en la actualidad, con esta herramienta se trabajó el lenguaje programación C#, lo que permitió tener un acercamiento sobre cómo los APIs son necesarios para el consumo de los recursos.

Se profundizó el conocimiento de cómo se gestiona una base de datos haciendo uso de un motor de bases de datos que en este caso fue postgreSQL, donde en la industria ha venido emergiendo gracias su facilidad de uso y su bajo consumo de recurso

Por otra parte es importante destacar el fortalecimiento en el conocimiento de diseño Web que conllevo este proyecto esto porque se debió implementar uso de tecnologías modernas Angular, Bootstrap, CSS y HTML5, los cuales dieron mayor criterio para pensar en el diseño basado en MVC(*model view Controller*)

Recomendaciones

En relación a la asignación de proyecto, se recomienda que en futuros enunciados se analice el entorno de los integrantes de los grupos que lo van a conformar, esto porque con respecto a los demás miembros con equipo completo, se le complica

menos las distribuciones de la tareas, porque por ese decaimiento extra de tareas perjudica a los miembros del equipo de desarrollo.

Ahora bien en relación a las habilidades desarrolladas y algunas que se puedan mejorar se tiene lo siguiente:

Se recomienda en lo que respecta al desarrollo de la móvil el uso de herramientas que faciliten el desarrollo haciendo uso del SDK del lenguaje pero con un IDEs que beneficie la facilidad de su uso y no sature las computadoras.

Se insta a seguir implementado el uso de de la programación web como herramienta del curso, esto porque como estudiante se logra enfrentar a problemas con una complejidad semejante o mayor a la del mercado laboral, y de esta forma la curva de aprendizaje va a ser de provecho y satisfacción.

Bibliografía Consultada para el Proyecto.

AngularJS (2019-02-05). Recuperado de: https://angular.io/

Bootstrap Themes & Templates (2019-02-05). Recuperado de: https://getbootstrap.com/docs/4.1/getting-started/introduction/

C# Coding Conventions (C# Programming Guide). (2018-10-04). Recuperado de: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-aprogram/coding-conventions

Hasan, J. (2004). *Expert Service-Oriented Architecture in C#*. Apress.

How to Write Doc Comments for the Javadoc Tool. (2018-10-04). Recuperado de: http://www.oracle.com/technetwork/articles/java/index-137868.html

RazorCX Technologies:
https://github.com/razorcx/json-example/blob/master/JsonExample/JsonExampleForm.cs

Singer, A. (2011). Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, *30*(1), 20-36.

SpotifyAppRemote (Spotify App Remote). (2020). Retrieved 21 November 2020, from https://spotify.github.io/android-sdk/app-remote-lib/docs/com/spotify/android/appremote/api/SpotifyAppRemote.html