Requisitos del proyecto

El departamento académico de CampusLands desea crear un programa que le permita llevar el seguimiento académico de todos los c***ampers*** que se encuentran matriculados en el programa intensivo de programación.

Para ello usted es contratado para liderar el desarrollo del programa que debe cumplir con las siguientes especificaciones:

1. El programa debe permitir a las personas encargadas de procesar las inscripciones a el programa; La información que se tiene por cada c***amper*** es la siguiente :

* # de identificación.
* Nombres.
* Apellidos.
* Dirección.
* Acudiente.
* Teléfonos de contacto(# de celular y #fijo).
* Estado (En proceso de ingreso, Inscrito, Aprobado,Cursando, Graduado, Expulsado, Retirado).
* Riesgo

1. Campus cuenta con diferentes rutas de entrenamiento las cuales deben cumplir los candidatos que superen la prueba inicial. Las rutas son las siguientes:

* Ruta NodeJS
* Ruta Java
* Ruta NetCore

1. El programa debe tener tres roles: Ca**mper, Trainer y Coordinador** , donde esta última persona debe contar con una opción en el programa que le permita registrar la nota de los c***ampers*** que se han registrado y con ello cambiar su estado a “**Aprobado**”. La prueba es aprobada si el promedio entre la nota teórica y la nota practica es mayor o igual a 60.
2. Campus cuenta con diferentes áreas de entrenamiento en la cuales los c***ampers*** aprenden los diferentes stacks tecnológicos dependiendo de las rutas de entrenamiento. Por el momento se cuenta con tres áreas de entrenamiento con una capacidad máxima de 33 ca**mpers (Tener en cuenta que hay clases cada 4 horas)**.
3. La **coordinación académica** desea poder crear nuevas rutas de entrenamiento las cuales contienen la siguiente información (módulos):

* Fundamentos de programación (Introducción a la algoritmia, PSeInt y Python)
* Programación Web (HTML, CSS y Bootstrap).
* Programación formal (Java, JavaScript, C#).
* Bases de datos (Mysql, MongoDb y Postgresql). Cada ruta tiene un SGDB principal y un alternativo.
* Backend (NetCore, Spring Boot, NodeJS y Express).

1. Los campe***rs*** que pasaron de “Ins**critos**” a “A**probados**” podrán ser asignados a cualquiera de las rutas que se han creado previamente. Se debe tener en cuenta que no se puede exceder la capacidad de cada una de las áreas de entrenamiento.
2. Ca**mpusLands** cuenta con **Trainers** expertos encargados de dirigir cada una de las rutas de entrenamiento. Esto quiere decir que a cada uno se le podrán asignar diferentes rutas de entrenamiento teniendo en cuenta su horario.
3. La **coordinación académica** desea contar con un módulo de matriculas que le permita asignar los ***campers*** aprobados, **trainer** encargado, ruta de entrenamiento asignada, fecha de inicio, fecha finalización y salón de entrenamiento.
4. Periódicamente los c***ampers*** son evaluados para conocer las habilidades adquiridas durante el proceso de entrenamiento, donde cuando finaliza cada modulo los ***campers*** deben presentar una prueba teórica y una prueba practica. Esta prueba es considerada como aprobada si el promedio de las dos dan un valor mayor o igual a 60. Aqui la prueba teórica tiene un peso de 30% y la prueba practica tiene un peso del 60%, donde durante dicho proceso el T**rainer** realizará quizes, trabajos los cuales tienen un peso del 10%. Al finalizar el proceso de evaluación se considerará aprobado el modulo si la nota final es mayor a 60.
5. La **coordinación académica** cuando finaliza cada uno de los módulos de las rutas evalúa el rendimiento de cada uno de los ca**mpers** teniendo en cuenta la nota obtenida en cada modulo. Si la nota es menor a 60 el **camper** queda en rendimiento bajo lo cual genera un llamado de atención. Por esto mismo, se deberá permitir consultar los **campers** que se encuentren en riesgo alto.
6. El módulo de reportes debe tener las siguientes funcionalidades:

* Listar los **campers** que se encuentren en estado de inscrito.
* Listar los **campers** que aprobaron el examen inicial.
* Listar los entrenadores que se encuentran trabajando con **CampusLands**.
* Listar los **campers** que cuentan con bajo rendimiento.
* Listar los **campers** y **trainers** que se encuentren asociados a una ruta de entrenamiento.
* Mostrar cuantos **campers** perdieron y aprobaron cada uno de los módulos teniendo en cuenta la ruta de entrenamiento y el entrenador encargado.

ANALISIS DEL PROYECTO

* ESTRUCTURA PARA LA SOLUCION

Se debe de crear 3 tipos de usuarios con 3 tipos de permisos:

camper (solo permisos de lectura y modificar información personal)

Trainer (permisos de escritura y de lectura con ciertas restricciones en algunos módulos)

Coordinador (permiso total o de administrador)

**Tipos de permisos especificados para cada usuario**

* Camper:
* Registrar información básica.
* Consultar su estado y asignaciones.
* Trainer:
* Dirigir rutas de entrenamiento.
* Registrar notas y evaluaciones de los campers.
* Coordinador:
* Crear rutas nuevas.
* Cambiar estados de campers ("Inscrito" a "Aprobado").
* Asignar rutas a campers y trainers.
* Generar reportes sobre el desempeño y estado de campers y trainers.

**DISEÑO DEL PROGRAMA**

**TIPO DE PROGRAMA: MODULAR** Se necesita dividir toda la estructura en subconjuntos pequeños y manejables para poder modificar el codigo de una manera más rápida y entendible

**ALMACENAMIENTO DE DATOS: JSON**

**Gestión de usuarios: Registro y actualización de información de campers, Registro de trainers.**

**Gestión de rutas: Creación de rutas por parte del coordinador, Asignación de campers a rutas trainers.**

**Evaluaciones: Registro de notas por módulo, Cálculo de promedios ponderados para aprobar módulos.**

**Reportes: Listado de campers y trainers según los criterios dados, Generación de reportes por rendimiento y estado.**

**Desarrollo de archivos**

**El proyecto estará organizado en los siguientes archivos**

**main.py: Archivo principal que ejecuta el programa.(** El archivo principal se encargará de presentar el menú interactivo y llamar a las funciones de los módulos.)

**modules/: Carpeta para archivos modulares. Ejemplo:**

**gestion\_usuarios.py: Gestión de campers y trainers.(** Aquí se implementan funciones para registrar campers y trainer)

**gestion\_rutas.py: Gestión de rutas.**

**evaluaciones.py: Gestión de evaluaciones y notas.**

**reportes.py: Generación de reportes.**

**data.json: Archivo JSON para almacenar los datos del programa.**