Universidad Juárez del Estado de Durango

Documentación acerca de los procesos de la plataforma de formación integral.

Manual

Plataforma Formación Integral

Índice

[**Back-end** 2](#_Toc94534553)

[**Aplicación Alumnos.** 5](#_Toc94534554)

[**Aplicación eventos** 9](#_Toc94534555)

[**Aplicación de FormacionIntegral** 15](#_Toc94534556)

[**Archivos de servicios** 19](#_Toc94534557)

[**AlumnosDataService.js** 19](#_Toc94534558)

[**EventosDataService.js** 20](#_Toc94534559)

[**FormacionInDataService.js** 22](#_Toc94534560)

[**Router** 23](#_Toc94534561)

[**Front-end** 25](#_Toc94534562)

[**Módulo de inicio.** 25](#_Toc94534563)

[**Módulo de eventos.** 27](#_Toc94534564)

[**Módulo de actualización de eventos.** 33](#_Toc94534565)

[**Módulo de creación de eventos.** 34](#_Toc94534566)

[**Módulo formación integral** 36](#_Toc94534567)

[**Módulo de registro.** 37](#_Toc94534568)

[**Módulo de asistencia.** 41](#_Toc94534569)

[**Módulo de validación de créditos.** 44](#_Toc94534570)

[**Módulo de alumnos para administradores.** 46](#_Toc94534571)

[**Módulo de historial de eventos.** 48](#_Toc94534572)

[**Moduló de calendario.** 53](#_Toc94534573)

[**Módulo de vista de alumnos.** 55](#_Toc94534574)

[**Moduló de evidencias.** 61](#_Toc94534575)

# **Back-end**

El back-end de nuestra plataforma se desarrolló con el framework de desarrollo web Django y su framework Django REST framework para la creación del API la cual proporcionara los datos indicados por las peticiones http los cuales seran obtenidos de nuestra base de datos en PostgreSql.

projectpfi/projectpfi/Settings.py

Debido al funcionamiento de Django se crearon tres aplicaciones para utilizar, estas son Alumnos, la cual es para manejar todos los datos acerca de los alumnos, eventos la cual funciona para obtener todos los datos de sus eventos y por último FormacionIntegral necesaria para los datos acerca de los alumnos registrados a sus eventos, estas aplicaciones deben ser importadas en el archivo principal de ajustes ubicado en el directorio principal projectpfi para poder utilizarlas, además de agregar el framework Django REST.

Captura de pantalla de computadora

Descripción generada automáticamente

Es importante que el archivo de ajustes tiene varias cosas importantes necesarias para el funcionamiento del sistema, como lo es definir el directorio base de nuestro proyecto importando la librería os, se define el directorio base para después definir las rutas para el almacenamiento de archivos y se define la ruta para las urls.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

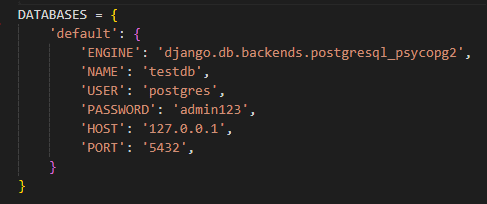
Captura de pantalla de computadora

Descripción generada automáticamente

Además de esto se añade una característica de Django REST Framework llamada django\_filters para poder filtrar los datos de un arreglo de datos proporcionado por el api creado, esta caracteriza se utilizó bastante para obtener datos específicos.

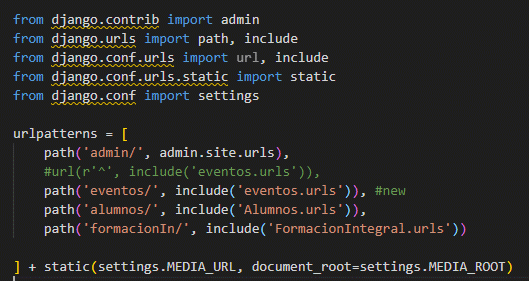


Por último, se configura la base de datos para utilizar la que es de preferencia, en este caso se utiliza una base de datos en PostgreSql, para más información de la instalación de otras bases datos consultar la documentación de Django, el parámetro Engine recibe el tipo de base de datos, name recibe el nombre de la base de datos, user recibe el usuario utilizado de PostgreSql y su respectiva contraseña, host recibe la url del servidor y su puerto.



projectpfi/projectpfi/Urls.py

Este archivo es el principal para las urls en el cual se importan las urls de cada aplicación por lo cual en este archivo registraremos una url por cada aplicación creada, en este caso tres, además de agregar una extra para el panel de administración. Por último, se agrega una ruta al final del archivo con la finalidad de poder obtener la ruta de las imágenes y poder mostrarlas en nuestro html, todo esto gracias a las rutas que definimos en el archivo de configuraciones.



A continuación, se muestra la estructura de cada aplicación separándolos por sus diversos archivos, los cuales se dividen en 7 pero solo utilizaremos 5.

El archivo admin.py es el encargado de registrar los modelos definidos de la aplicación al panel de administración en el cual podemos llevar a cabo diversas acciones para mostrar, modificar, eliminar y agregar registros desde el panel de administración.

El archivo models.py sirve para definir un modelo de datos en nuestra aplicación, este modelo se podría decir que es la estructura de las tablas de nuestra base de datos, por lo cual aquí se definen los campos, con su respectivo nombre, el tipo entre otras configuraciones. Una vez que ya se tengan los modelos es necesario aplicar las migraciones y luego migrar los datos para ver los cambios en la base de datos.

El archivo serializers.py es el encargado de interpretar los modelos de la aplicación en un formato Json para poder utilizarlos como si fuera una API mediante Django REST framework, solo importamos el modelo y definimos todos los campos del modelo que es importante que coincidan con los nombres del modelo.

El archivo urls.py permite agregar o definir las urls de nuestra aplicación, por lo cual se define la ruta y añade la opción de agregar el método que se ejecutara una vez que se acceda a la url definida, por lo que hay que importar los métodos del archivo views.py para utilizarlos en las rutas, es importante decir que por cada método debe existir una url o por lo menos es lo recomendado.

El archivo views.py es el encargado de manejar la lógica de los datos, entonces este archivo es donde se crearán los métodos y funciones necesarias para obtener, crear, actualizar, eliminar y buscar los datos de nuestra base de datos, estos métodos son los que se importan en las urls para ejecutarlos y se lleve a cabo el proceso indicado o definido desde el archivo views.py.

A continuación, se muestra la estructura de cada aplicación mostrando los 5 archivos anteriormente mencionados.

## **Aplicación Alumnos.**

Projectpfi/Alumnos/models.py

En la aplicación de Alumnos definimos el siguiente modelo con los datos mostrados, primero definiendo una clase la cual es el nombre de nuestra tabla para después agregar los campos utilizando el nombre del campo, seguido del tipo utilizando la variable models importada al inicio y ya dentro añadiendo un verbose name para identificar el campo por buenas prácticas, seguido de una longitud de caracteres, es importante mencionar que cada tipo de campo recibe diversos parámetros por lo que es necesario ver la documentación de los modelos en Django para mas información.

Captura de pantalla de computadora

Descripción generada automáticamente

Project/Alumnos/serializers.py

Una vez definimos nuestro modelo y ejecutamos las migraciones debemos de configurar el intérprete de nuestros datos a formato Json para utilizar los datos como una API, es necesario agregar este archivo al directorio debido a que cuando se crea la aplicación no viene por defecto.

Se importan las librerías necesarias al inicio como son los serializers desde rest\_framework seguido del modelo de alumnos. Se crea una clase la cual es la que utilizaremos en nuestro archivo de views.py para obtener los datos, dentro de esa clase añadimos otra llamada meta en la cual añadimos el modelo y sus campos. Se pueden crear varias clases para obtener diversos datos.

Captura de pantalla de computadora

Descripción generada automáticamente

Projectpfi/Alumnos/views.py

Lo primero es importar lo necesario para manejar los datos, como lo son el modelo, el serializer y las vistas genéricas de rest\_framework la cual ya tiene incluidas los métodos básicos para crear, actualizar, obtener y eliminar datos, después de esto se procede a realizar los métodos.

Captura de pantalla de computadora

Descripción generada automáticamente

Las vistas genéricas de rest\_framework tienen como objetivo ejecutar las consultas más comunes como los son obtener, actualizar, crear y eliminar, solo es necesario importar la vista deseada dentro los parámetros que recibe la clase, esta clase es la que vamos a añadir en la url del archivo urls.py, lo único que se necesita es añadir la variable queryset la cual obtendrá todos los datos de nuestro modelo, además de agregar la variable serializer\_class pasandole el serializer que definimos. Estas dos variables son propias de rest\_framework por lo que es necesario que tenga estos nombres si no puede ocasionar errores.

AlumnosCreate

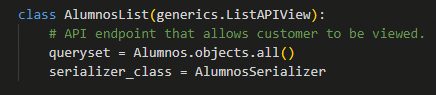
Este método es el encargado de crear un registro en la base de datos, en este caso pasamos como parámetro las vistas genéricas el CreateAPIView el cual ejecuta el proceso de un POST para insertar los datos en nuestra base de datos. Dentro de la clase definimos el queryset con el modelo y el serializer indicado debido a que si no coinciden los campos pueden haber errores.

Captura de pantalla de computadora

Descripción generada automáticamente

AlumnosList

Este método es el encargado de obtener los datos de todos los registros existentes en la tabla de nuestra base de datos, por lo cual este obtendrá a todos los alumnos del sistema. Se define el nombre de la clase y pasamos como parámetro la vista genérica ListAPIView la que ejecuta un GET para obtener los datos de todos los alumnos, al final añadimos el modelo y su serializer.



AlumnosDetail

Este método es para obtener los datos de un registro en específico en nuestra base de datos, se define el nombre de la clase y pasamos como parámetro la vista genérica RetrievAPIView, esta vista genérica especifica que debe recibir un parámetro que es el id, por lo que al momento de crear la url de esta clase es necesario crear una ruta que reciba como parámetro la pk del alumno. Esta vista ejecuta un método GET para recibir los datos, dentro de la clase añadimos el modelo y el serializer.

Captura de pantalla de computadora

Descripción generada automáticamente

AlumnosUpdate

Este método es el encargado de actualizar un registro especifico de nuestra base de datos, se define el nombre de la clase y pasamos la vista genérica RetrieveUpdateAPIView la cual especifica que debe recibir como parámetro el id del registro en este caso el id del alumno por medio de la url para actualizar los datos ejecutando un método PUT, dentro de la clase añadimos el modelo y el serializer.

Captura de pantalla de computadora

Descripción generada automáticamente

AlumnosDelete

Este metod es el encargado de eliminar un registro especifico de nuestra base de datos, definimos el nombre de la clase y pasamos como parámetro la vista genérica RetrieveDestroyAPIView la cual especifica que debe recibir como parámetro el id del alumno por medio de la url para eliminar los datos por medio de un método DELETE.

Captura de pantalla de computadora

Descripción generada automáticamente

Projectpfi/Alumnos/urls.py

Se importan las vistas desde el archivo view.py y agregamos la variable path desde django.urls y dentro del arreglo urlpatterns debemos agregar las rutas necesarias, iniciando con la variable path, seguido de la ruta que queremos además de verificar si necesitamos una ruta dinámica, debido a que lo siguiente es agregar el método que queremos ejecutar seguido del método .as\_view(), es importante recordar que algunos necesitaban la primary key del registro y al final añadimos un nombre a la ruta.

Captura de pantalla de computadora

Descripción generada automáticamente

Projectpfi/Alumnos/admin.py

Aquí solo debemos de registrar el modelo utilizando el decorador @admin.register() que recibe como parámetro el modelo, seguido de eso creamos una clase la cual tiene varias opciones para poder mostrar los datos, lo cual se hace con la variable list\_display donde agregamos el nombre de los campos que queremos mostrar en el panel, estos deben de coincidir con los nombres del modelo. La variable list\_editable se encargar de ver los campos que se pueden editar desde el administrador mientras que la variable list\_filter funciona para filtrar los datos mediante los campos que elijas agregar. Existen más funcionalidades para personalizar este panel, por lo que hay que verificar la documentación de Django para obtener más información.

Captura de pantalla de computadora

Descripción generada automáticamente

A continuación, se mostrarán las dos aplicaciones faltantes, pero realmente tienen características muy parecidas ya que se utilizan las vistas genéricas así que para no ser redundante se explicara, solo las cosas que no se hayan explicado.

## **Aplicación eventos**

Projectpfi/eventos/models.py

Se importan las librerías necesarias. En este caso se añade una nueva llamada CASCADE la cual se utiliza para eliminar registros vinculados a una llave primaria de forma automática una vez que se borra ese registro, se utiliza para el uso de llaves foráneas.

Captura de pantalla de computadora

Descripción generada automáticamente

Se definió el modelo para los datos de los eventos llamado eventos.

Captura de pantalla de computadora

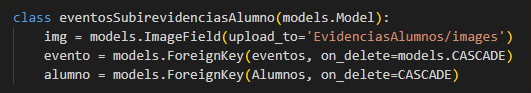
Descripción generada automáticamente

No obstante, en esta aplicación tenemos la inclusión de otros dos modelos, uno dedicado al calendario de la aplicación el cual es necesario para hacer funcionar el componente que recibe esta información en el Front-end con la siguiente estructura de datos, name muestra el nombre del evento, color muestra el color en el que se pintara el evento, start es cuando inicia el evento y end es cuando termina, details muestra la descripción del evento, por ultimo tenemos la llave foránea que está vinculada a nuestra llave primaria del modelo de eventos, la cual recibe el modelo y el parámetro on\_delete para eliminar registros vinculados a este.

Captura de pantalla de computadora

Descripción generada automáticamente

Por último el modelo para subir las evidencias de los alumnos, las cuales tienen un campo ImageField el cual recibe como parámetro la variable upload\_to que sirve para definir una ruta para almacenar las imágenes, la cual primero es necesario configurar la MEDIA\_URL desde el archivo de configuraciones pero eso ya está configurado, básicamente aquí decimos que en la carpeta media definida anteriormente crea un carpeta llamada EvidenciasAlumnos y dentro de esa crea la de images donde se almacenaran todos los archivos. Por último, este tiene dos llaves foráneas.



Projectpfi/eventos/serializers.py

Se crearon los siguientes serializers. eventosSerializer el cual es utilizado para todos los métodos para crear, editar, eliminar, actualizar y obtener los datos de los eventos.

Captura de pantalla de computadora

Descripción generada automáticamente

calendarioSerializer es utilizado para todas las consultas relacionada al módulo de calendario para obtener los datos de las fechas, nombres, colores, el inicio y el fin para poder plasmarlos en el módulo de calendario.

Captura de pantalla de computadora

Descripción generada automáticamente

Por último, se tuvieron que utilizar dos serializers distintos para las evidencias, debido a que tenemos la inclusión de un campo de tipo imagen se optó por utilizar un serializer para crear el registro debido que al utilizar uno con todos los campos causaba un error al crearlo debido a que no contamos con el id ya que este es por default mientras que con los demás solo se usó el que tiene todos los campos.

Texto

Descripción generada automáticamente

Projectpfi/eventos/views.py

Se importaron las siguientes variables para utilizar en los métodos.

Captura de pantalla de computadora

Descripción generada automáticamente

Para los eventos se utilizaron puras vistas genéricas y métodos similares a la aplicación de Alumnos, las uncias diferencias es que cambiasmos el modelo utilizado en este caso eventos y su respectivo serializer eventosSerializer, además de agregar otra variable llama filter\_backends la cual agregamos o importamos desde nuestro archivo de configuraciones en nuestro apartado de REST\_FRAMEWORK, solo hay que importarlo desde esa variable, básicamente este campo lo utilizamos en los métodos GET para poder filtrar los datos mediante los campos que se agreguen en el arreglo de nuestra variable filter\_fields, funciona como si pasáramos parámetros mediante la url, en este caso podemos filtrar los datos obtenidos y tener datos más exactos.

Un ejemplo de esto es, url?categorias=Intelectuales los datos que retornar nuestro GET seran todos aquellos que concuerden con la categoría Intelectuales. Esto se usará desde los archivos de servicios al momento de realizar las peticiones.

Captura de pantalla de computadora

Descripción generada automáticamente

El calendario solo utiliza dos vistas genéricas para crear y obtener, con su respectivo modelo y serializer además de también incorporar los filtros al obtener los datos para buscar eventos por su fecha.

Captura de pantalla de computadora

Descripción generada automáticamente

Esta clase es para crear el registro de evidencia con su respectiva imagen, primero verifica que sea un método POST y recibe los datos de la peticion, después valida que la imagen se encuentre en la petición en dado caso retorna un mensaje de error. Después utiliza el serializer para guardar los datos de la petición, una vez que guardo los datos verifica si es valido para convertirlo y guardarlo. Este método utiliza el modelo

Captura de pantalla de computadora

Descripción generada automáticamente

Este método primero recibe como parámetro la llave primaria del registro de la evidencia que se quiere actualizar, mediante la función get\_object se obtienen los datos de la evidencia mediante la llaver primaria y retorna sus datos. Entonces cuando se ejecute el método PUT se ejecuta la funcion get\_object para obtener los datos. En nuestro serializer añadimos los datos actuales y pasamos como segundo parámetro los nuevos datos que vienen de la petición, se verifica si el serializer es válido, si es válido se validara y seguido de eso actualizaremos mediante el método .save pasando como parámetros update\_fields y el campo a modificar en este caso img, luego obtenemos nuevamente los datos para verificar que los cambios fueron correctos mediante un mensaje. En este caso utilizamos el modelo de eventoSubirevidenciasAlumno y el serilizer de evidenciaSerializer.

Texto

Descripción generada automáticamente

Por último, tenemos una vista genérica para obtener los datos, esto se decidió así para hacer uso de los filtros para la obtención de datos en el módulo de evidencias, que en su petición http recibirá como parámetro el id de evento y el alumno para encontrar las evidencias del evento.

Texto

Descripción generada automáticamente

Projectpfi/eventos/urls.py

Captura de pantalla de computadora

Descripción generada automáticamente

ProjectPfi/eventos/admin.py

Se registraron los tres modelos al panel de administración como se mostró anteriormente.

Captura de pantalla de computadora

Descripción generada automáticamente

## **Aplicación de FormacionIntegral**

Projectpfi/FormacionIntegral/models.py

Este es el módelo utilizado para esta aplicación, este tiene dos llaves foráneas, evento y alumno, por lo que se importaron los modelos necesarios para su uso.

Captura de pantalla de computadora

Descripción generada automáticamente

Projectpfi/FormacionIntegral/serializers.py

En esta aplicación existen 2 serializers uno para la obtención de los datos del modelo el cual se llama formacionInSerializer la que tiene definidos los campos de nuestro modelo tal cual se creo en el archivo de models.py, no obstante, en este modulo debemos obtener acceso a los datos de las llaves foráneas por lo que se creó un segundo seializer.

Captura de pantalla de computadora

Descripción generada automáticamente

Este serializer fue llamado FormacionInEventoSerializer el cual se ve con una estructura diferente, debido a que para obtener los datos de la llave foránea primero debemos crear las variables y añadirlas haciendo tarjet a la variable foránea de la cual necesitamos obtener los datos, en este caso evento. Por lo cual se define la variable y utilizando el serializer definimos el tipo de campo y dentro hacemos el tarjet de los datos utilizando source, es importante que los nombres deben coincidir con los del modelo sin no ocasionara errores.

Captura de pantalla de computadora

Descripción generada automáticamente

Una vez que ya terminamos de obtener los datos debemos de añadirlos en los campos de la clase meta para poder hacer uso de ellos.

Captura de pantalla de computadora

Descripción generada automáticamente

Projectpfi/FormacionIntegral/views.py

Se utilizaron las vistas genéricas por de rest\_framework, de los cuales todos tienen el respectivo modelo y su serializer, además de contar con los filtros en los métodos GET para obtener datos más precisos, además de mencionar que el método FormacionInEventosList es el único que recibe el serializer FormacionInEventosSerializer.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Projectpfi/FormacionIntegral/urls.py

Las rutas indicadas de los eventos, algunas reciben el id o llave primaria del registro para funcionar, y la url ListEventsData es la que muestra los datos de la llave foránea.

Captura de pantalla de computadora

Descripción generada automáticamente

Projectpfi/FormacionIntegral/admin.py

Captura de pantalla de computadora

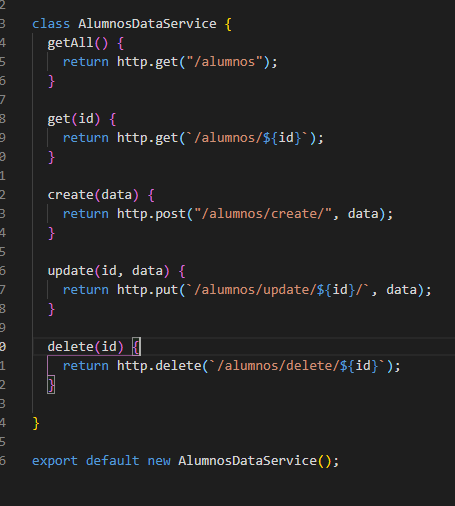
Descripción generada automáticamente

# **Archivos de servicios**

Estos archivos son los encargados de conectar el back-end y el front-end mediante el uso de peticiones http las cuales van a generar una respuesta de dependiendo de los datos proporcionados.

Estos archivos están ubicados en la ruta pfi-front/src/services.

## **AlumnosDataService.js**



Retorna los datos de un delete para eliminar a un alumno utilizando el id del alumno.

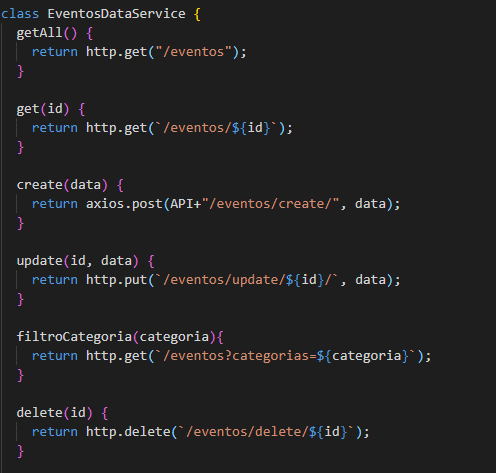
Retorna los datos de un put para actualizar los datos del alumno, utiliza el id y sus datos actualizados.

Retorna los datos de un post para crear un alumno, utiliza los datos del alumno en la variable data.

Retorna los datos de un alumno en específico, utilizando el id.

Retorna los datos de todos los alumnos.

## **EventosDataService.js**



Retorna los datos de un delete para eliminar un evento por medio de su id.

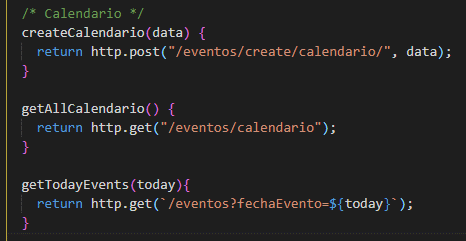
Retorna los datos de todos los eventos creados con la categoría que se especifique.

Retorna los datos de un put para actualizar los datos del evento por medio del id y los datos del evento.

Retorna los datos de un post el cual crea un evento por medio de los datos proporcionados.

Retorna los datos de un evento en especifico usando el id del evento.

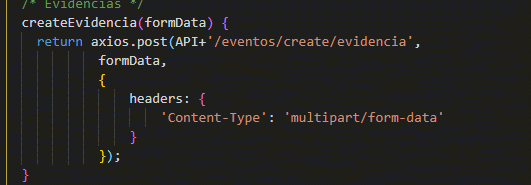
Retorna los datos de todos los eventos.



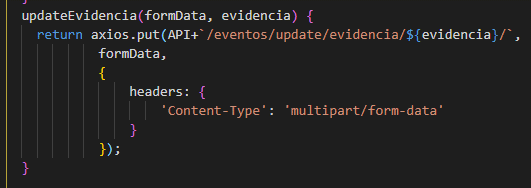
Retorna los datos de todos los eventos de una fecha específica en este caso para los eventos del día.

Retorna los datos de todos los eventos registrados en el calendario.

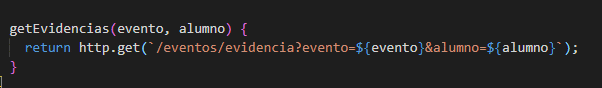
Retorna los datos de un post para agregar o crear un evento en el calendario, usando los datos del evento.



Se utiliza para hacer un post el cual crea la evidencia con sus datos y respectiva imagen, recibiendo como parámetro el formData que recibe del Front-end.

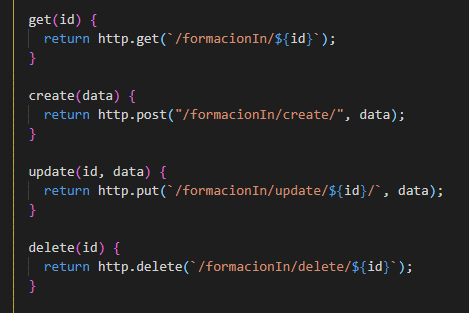


Se utiliza para actualizar los datos de una evidencia, por lo que utiliza el método put. Recibe como parámetros el formData desde el Front-end y el id de la evidencia.



Este método se utiliza para obtener la evidencia de un alumno especifico, recibe como parámetros el id del evento y el id del alumno, retornando los datos coincidentes.

## **FormacionInDataService.js**

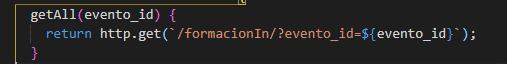


Retorna los datos de un delete para eliminar el registro de un alumno a un evento, utiliza el id de registro.

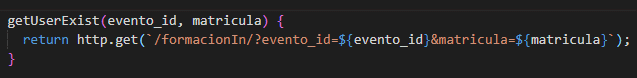
Retorna los datos de un put para actualizar los datos del registro, en este caso se usa para la asistencia. Utiliza el id del registro y sus datos.

Retorna los datos de un post para crear el registro de un alumno a un evento. Utilizando sus datos.

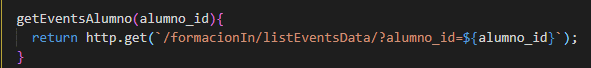
Retorna los datos del alumno registrado al evento, mediante su id de registro.



Se utiliza para retornar los datos de todos los alumnos registrados a cierto evento, recibe el id del evento.



Se utiliza para obtener los datos de un alumno registrado a un evento, recibe como parámetros el id del evento y la matricula del alumno con el fin de verificar si un alumno ya se encuentra registrado a un evento, así evitando el duplicado de datos.



Se utiliza para obtener los datos de todos los eventos en los que se encuentra registrado un alumno, recibe como parámetro el id del alumno.

# **Router**

El router es el encargado de manejar las rutas de nuestra página web, el cual funciona con rutas estáticas o rutas dinámicas. Las rutas estáticas no reciben ningún parámetro en su url por lo cual las dinámicas si, estas se utilizan agregando dos puntos (:) y el nombre de el parámetro dinámico.

Las rutas se definen utilizando el parámetro path, este indica la ruta en la url que se necesita para acceder a esta, name sirve para identificar el nombre del componente y component que es la ruta en nuestro proyecto donde se encuentra el componente que se va a mostrar en el html.

Rutas para Eventos.

Texto

Descripción generada automáticamente

Módulo de inicio

Módulo de eventos

Actualizar eventos

Registrar eventos

Módulo de calendario

Rutas para Formación Integral.

Texto

Descripción generada automáticamente

Módulo de validación C.

Módulo de asistencia

Módulo de registro

Módulo de Formación integral

Texto

Descripción generada automáticamente

Módulo de historial de eventos. Admin.

Módulo de vista de alumnos.

Módulo de evidencias.

Módulo de alumnos Ad.

# **Front-end**

Nuestro Front-end esta desarrollado en un proyecto con vue el cual utilizaremos para conectar al back-end y obtener datos mediante los archivos de servicios, además de utilizar el framework vuetify para el desarrollo del mismo.

## **Módulo de inicio.**

Este módulo se ubica en la carpeta de pfi-front/src/components.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Este módulo es donde todo usuario al acceder a la plataforma vera, por eso su nombre de módulo de inicio o página principal. Este módulo cuenta con un carrusel el cual muestra diversas imágenes acerca de la institución. Donde el funcionamiento depende de las directivas establecidas por el framework vuetify acerca de este componente, al cual debemos de pasar un arreglo en este caso Items el cual tenga la url o ruta de las imágenes que se muestran, donde se utiliza un ciclo v-for para mostrar todas las imágenes gracias a la directiva src que es la encargada de seleccionar la imagen.

Captura de pantalla de computadora

Descripción generada automáticamente Captura de pantalla de computadora

Descripción generada automáticamente

Además de esto muestra un pequeño apartado donde muestra las funciones del área de formación integral, después de eso salen una serie de botones donde podemos dar clic e inmediatamente aparecerá un nuevo apartado donde muestra información acerca del botón seleccionado.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Cada botón cuenta con la directiva @click para ejecutar métodos al darle clic, este ejecuta el método selected() el cual recibe un numero ya sea 1, 2, 4. La información de los botones esta oculta por la directiva v-if, la cual dependerá del número seleccionado por el botón, el cual mostrara la nueva información además de guiarte directamente a ella gracias al método de scrollView().

Captura de pantalla de computadora

Descripción generada automáticamente Captura de pantalla de computadora

Descripción generada automáticamente

## **Módulo de eventos.**

Este módulo se ubica en la carpeta de pfi-front/src/components/Eventos donde están todos los componentes involucrados en este módulo.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

La vista principal al ingresar a este módulo es la encargada del componente EventosList.vue donde existen 3 partes importantes para el componente, la primera es la barra de filtros por categoría de los eventos, de los cuales tenemos 6 opciones, donde puedes listar todos los eventos a mostrar o dependiendo de la categoría que elijas, hasta listar los eventos externos.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Cada icono o boton de la barra cuanta con la directva @click la cual ejecutara el metodo necesario para hacer la consulta de los eventos, existen 2 metodos para esto uno es el que lista todo los eventos llamado retrieveEventos() y el otro llamado FiltroCategorias() el cual recibe un parametro de tipo String para realizar la busqueda por la categoria indicada.

Captura de pantalla de computadora

Descripción generada automáticamente

Estos métodos ejecutan una función que hace una petición http al back-end desde la ruta pfi-front/src/services/EventosDataService.js, en la que se encuentran todas las funciones para conectar el back-en y el front-end los cuales retornan una respuesta mediante la variable de EventosDataService importada al inicio del script la cual nos permitirá acceder a todos los métodos declarados en este archivo. Al retornar una respuesta la interpretaremos de manera satisfactoria o no satisfactoria, de manera que si fue satisfactoria tendremos una respuesta con los datos que necesitamos en este caso los eventos, por lo cual los guardaremos en nuestra variable declarada en el apartado de data, la cual es la encargada de almacenar los eventos después de una petición para poder acceder a ellos y mostrarlos en el recuadro de eventos.

Captura de pantalla de computadora

Descripción generada automáticamente

En el recuadro de eventos cambiara dependiendo de los datos que tenga la variable eventos como ya se mencionó antes, este recuadro cuenta con la funcionalidad de un check box que te permite seleccionar uno de los eventos gracias al funcionamiento del componente v-data-table del framework vuetify, además de permitir la búsqueda de eventos, todo esto gracias a los props que recibe este componente.

Al utilizar la directiva v-model podemos vincular los datos del componente a una variable en este caso selected, debido a que estamos activando show-select la cual habilitara para cada uno de los datos un check box que al activarlo retornara los datos del item seleccionado gracias al v-model.

Es importante decir que tenemos activada la selección individual mediante el single-select con la finalidad de evitar utilizar datos equivocados. El prop de search es el asignado del buscador al cual solo debemos de agregar una variable que se encargue de guardar los datos del input ya que el componente hara la busqueda automaticamente.

Texto

Descripción generada automáticamente

Lo más importante de este componente es importante pasar como props los headers o encabezados de la tabla así como el contenido con el prop de ítems donde utilizamos la variable de eventos la cual ya se mencionó anteriormente, mientras que los encabezados es importante añadirlos mediante un arreglo de objetos en el cual cada objeto representa un encabezado, en el cual este contara con dos valores text que sería el texto que mostrara el encabezado y value que será el tarjet de los datos recibidos por el prop de items para representar el valor, por lo que hay que conocer como están guardados los datos desde que se hace la petición http la cual viene del back-end. También existen otras opciones que se pueden añadir al objeto para poder filtrar o incluso alinear.

Captura de pantalla de computadora

Descripción generada automáticamente

Ya una vez seleccionado el item el componente automáticamente cambiara el valor de la variable selected, la cual es utilizada para obtener los datos de cada evento mediante la selección de este, el componente guarda los datos en la variable como si fuera un arreglo debido a la posibilidad de tener una multi selección, por lo cual para acceder a los datos en el recuadro de datos de evento debemos de utilizar la variable en su posición 0 para acceder a los datos (selected[0].nombre).

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente Captura de pantalla de computadora

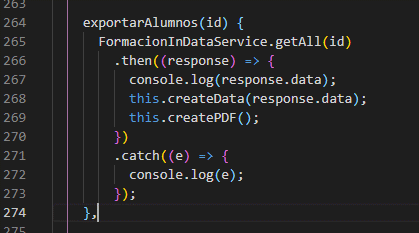
Descripción generada automáticamente

Este recuadro está dividido en dos ya que depende de una condicional para saber qué es lo que se mostrara en pantalla, ya sea la un recuadro de carga o la pantalla de los datos, esto dependerá de si la variable selected[0].tituloEvento es diferente de ‘’ ya que por defecto se inicializo así desde el inicio, por lo que al cambiar la variable tendrá un valor diferente que mostrara los datos del evento.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ese recuadro también ofrece la posibilidad de editar y crear una lista con los alumnos inscritos al evento en formato Pdf. Este botón funciona mediante el método exportarAlumnos() el cual recibe como parámetro el id del evento el cual se encuentra en la variable selected[0].id ya que este es el identificador único para cada evento que se ha creado. Si la petición es correcta recibiremos todos los alumnos recibidos al evento gracias al método getAll() exportado desde la carpeta de servicios el cual recibe el id del evento.



A su vez al recibir los datos correctos se ejecutarán dos métodos necesarios para crear el pdf con el listado de alumnos registrados al evento, así como los datos de este. Por lo cual el primer método llamado createData() el cual recibe los datos de todos los alumnos registrados al evento, entonces inmediatamente utilizando la variable response con la cual ejecutamos la petición pasamos directamente los datos al método createData() en su propiedad de data donde se guardan los datos de la petición.

El método createData() tiene la función de crear un arreglo con datos necesarios los cuales seran utilizados para crear el pdf, en este caso sustituye el valor de la asistencia por un string y guarda los datos para insertarlos en un arreglos mediante un bucle for-of.

Una vez que ya tenemos los datos necesarios para crear el pdf se ejecuta el método de createPdf() el cual para funcionar necesita el arreglo de los eventos el cual muestra el recuadro de los eventos y el arreglo de los alumnos creado por el metodo createData().

Captura de pantalla de computadora

Descripción generada automáticamente

El metodo de createPdf() funciona con el uso de la librería jspdf y un componente llamado jspdf-auto table para crear tablas utilizando la misma librería, este componente recibe un arreglo que son los datos que se van a utilizar y los header que son los encabezados de tablas, las cuales se usan en el pdf para mostrar información y son representados como columns en el componente.

Captura de pantalla de computadora

Descripción generada automáticamente

Primero se define la variable doc donde se define la orientación, el formato y el tamaño del pdf, para después ingresar de forma manual los saltos de línea y textos que se ubicaran en el archivo, después agregando la respectiva tabla en este caso la de evento.

Despues se realiza el mismo proceso para insertar una segunda tabla con los datos de los alumnos solo hay que cambiar el parametro body con los datos que vamos a utilizar, así como definir las columnas que funcionan de forma similar al componente v-data-table, solo que aquí el header es el texto que se mostrara y la dataKey seria el tarjet de los datos el cual mostrara la información que este dentro del arreglo.

Captura de pantalla de computadora

Descripción generada automáticamente

Al final se agrega un pie de pagina y al final se guarda el archivo con su respectivo nombre y extension para obtener la descarga del archivo al dar clic al boton de pdf.



Lo siguiete es el boton de editar, ubicado a un lado del boton de Pdf,­ este se encargar de modificar los eventos que ya se han creado para corregir alguna equivocación al momento de crear el evento. Este solo tiene un metodo que redirecciona a otra pantalla mediante el uso del router donde se muestra un formulario para realizar la edición del evento. Solo hay que mandar como parametro la direccion del router más el id del evento para acceder a el, ya que esta ruta es dinamica.

Captura de pantalla de computadora

Descripción generada automáticamente

Esta pantalla es el componente de para la edicion de eventos y se ubica en la ruta de pfi-front/src/components/Eventos/Eventos.vue.

## **Módulo de actualización de eventos.**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Al contar con un ruta dinamica es mas facil acceder a los datos debido a que ya tenemos acceso al id del evento por medio de los parametros de la url que mandamos desde el componente de eventos. Por lo cual al cargar los datos del componente este ejecuta una petición hhtp para obtener los datos del evento en especifico mediante el metodo getEvento() el cual recibe los parametros de la url para realizar la peticion gracias a la importacion del archivo de servicios EventosDataService donde llamamos el metodo get() para un evento en especifico. Una vez recibida la peticion se procede a guardar los datos del evento en una variable llamada currentEvento.

Captura de pantalla de computadora

Descripción generada automáticamente

Una vez con los datos almacenados solo hay que usar la directiva v-model para vincular los datos del evento a sus respectivas etiquetas html para mostrar y modificar los datos.

Captura de pantalla de computadora

Descripción generada automáticamente

Al final se encuentra un botón que tiene como objetivo ejecutar el método updateEvento() el cual tomara los datos de nuestro currentEvento, el cual ejecuta la petición llamada update para actualizar el evento pasando como parámetros el id del evento y los datos actualizados del evento, los cuales serán currentEvento ya que este está vinculado a las etiquetas html, así que si existe algún cambio en los datos este será reflejado al dar clic en el botón.

Captura de pantalla de computadora

Descripción generada automáticamente

## **Módulo de creación de eventos.**

Ubicado en la ruta pfi-front/src/components/Eventos/AddEventos.vue

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Este módulo funciona básicamente igual que el de actualizar eventos ya que funciona con la misma lógica, la única diferencia esta que en lugar de obtener los datos de un evento para modificarlos este los va a crear. Por lo que vincularemos nuestras etiquetas html a los datos en una variable llamada evento la cual será un objeto ya que definiremos el nombre de cada dato del objeto para que coincida con los valores de nuestra base de datos.

Captura de pantalla de computadora

Descripción generada automáticamente

Al dar clic en el botón de crear evento este ejecutar el método crear evento, donde obtiene los datos de las etiquetas html y los guarda en una variable llamada data la cual es pasada como parámetro en la petición para crear el evento llamada créate del archivo de servicios EventosDataService con el que se lleva a cabo el registro del evento. Y una vez que se creó correctamente el evento se ejecutara un segundo método para registrar el evento al módulo de calendario. El cual recibirá como parámetro los datos guardados en la variable data, pero aquí se hizo directamente con los datos de la petición que serían los mismos.

Captura de pantalla de computadora

Descripción generada automáticamente

Se guardan los datos en una variable llamada campos la cual es creada con la estructura necesaria para el funcionamiento del calendario en su respectivo módulo, donde al final se ejecuta una petición para registrar el evento en el calendario llamada createCalendario pasándole la variable campos como parámetros.

## **Módulo formación integral**

Ubicado en la ruta pfi-front/src/components/FormacionIntegral/FormacionIntegral.vue

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Este módulo es igual al de eventos la única diferencia es en el recuadro que muestra los datos del evento, este en lugar de ofrecer la opción de editar y crear un Pdf hay un botón que dice registro y asistencia, este ejecuta un método que lleva a otra pantalla el cual recibirá como parámetro el id del evento debido a que es una ruta dinámica con la finalidad de obtener datos más fáciles. Es importante destacar que este último recuadro se creó como un componente a parte el cual recibe los datos del evento mediante un arreglo con los datos a través de los props por lo que si se desea modificar u optimizar es necesario modificar el archivo infoEvent.vue ubicado en components/Eventos.

Captura de pantalla de computadora

Descripción generada automáticamente

## **Módulo de registro.**

Ubicado en la ruta pfi-front/src/components/FormacionIntegral/Registro.vue

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Este módulo cuenta con 2 apartados, uno que muestra una tabla con los alumnos disponibles, así como un buscador el cual sirve para encontrar alumnos de forma más fácil. Este recuadro utiliza la misma funcionalidad que los anteriores ya que se hace uso de un v-data-table, la única diferencia con los anteriores es que contamos con la propiedad de multi selección al cambiar la variable single-select a falso.

Captura de pantalla de computadora

Descripción generada automáticamente

Como ya se mencionó antes las variables necesarias para utilizar este componente son, selected la cual es la encargada de almacenar los datos seleccionados, headers los cuales son los títulos de los encabezados y los que identificar el tarjet de los datos que se mostraran, estos datos son obtenidos mediante la variable alumnos.

Captura de pantalla de computadora

Descripción generada automáticamente

La variable alumnos obtiene los datos al iniciar o entrar en la pantalla mediante el método retrieveAlumnos, este se ejecuta una petición al método getAll() del archivo externo AlumnosDataService importado para realizar las peticiones de los alumnos.

Captura de pantalla de computadora

Descripción generada automáticamente

El segundo apartado muestra el nombre del evento, el cupo total del evento y los cupos disponibles al momento, además de mostrar una tabla de los alumnos seleccionados para el registro al evento que al dar clic registrara a los alumnos, no sin antes verificar si ya están registrados o no hay cupos disponibles.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

El segundo recuadro está directamente relacionado con el primer recuadro, ya que se está utilizando un v-data-table el cual recibe como parámetros la variable headerRegistrados para mostrar los encabezados y el tarjet, mientras que en el parámetro de ítems que es de donde se obtinen los datos es la variable selected, la cual vinculamos al v-data-table de los alumnos. Esto permite que todos los alumnos seleccionados aparezcan en la nueva tabla. Para más información acerca de las propiedades v-slot es necesario verificar la documentación de vuetify específicamente en el componente utilizado.

Captura de pantalla de computadora

Descripción generada automáticamente

Al dar clic en el botón de registrar alumnos se ejecutara el método de validarAlumnos()

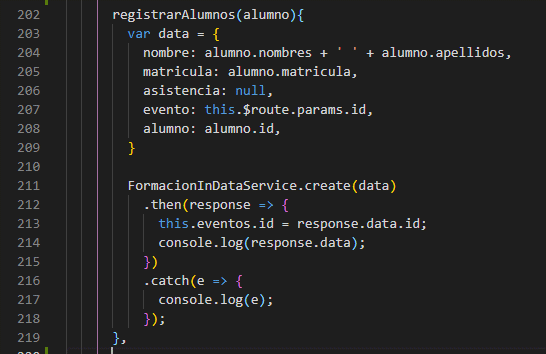
Captura de pantalla de computadora

Descripción generada automáticamente

Primero se ejecuta el método obtenerDisponible() este verifica que existan lugares disponibles en el evento, este numeró se guarda en la variable disponible la cual es utilizada en la condicional para evitar exceder el límite de usuarios registrados, si cumple la condicional se ejecutara un bucle for of, este bucle recibe la variable selected donde se encuentran almacenados los alumnos seleccionados y gracias a este tipo de bucle renombramos cada item del arreglo con el nombre de alumno.

Por cada iteración del ciclo se ejecutará una petición http desde la variable FormacionInDataService la cual se importó desde el archivo exterior de servicios FormacionInDataService.js, este archivo es el encargado de los alumnos registrados a los eventos, por lo cual se utilizó el método getUserExist el cual recibe como parámetros el id del evento y la matricula del alumno.

Al recibir los datos de la petición recibiremos un arreglo de objetos el cual tiene una propiedad llamada lenght, si esta propiedad recibe un valor de 0 significa que actualmente no existe un alumno con esa matricula registrado al evento, mientras si es un 1 es porque ya existe, entonces dependiendo de esta condicional se ejecutar el método de registrarAlumnos() recibiendo como parámetro los datos del alumno, en este caso la variable alumno especificada en nuestro ciclo.



Este método toma los datos del alumno y crea un arreglo de objetos con la sintaxis indicada por nuestro modelo de base de datos para llevar a cabo un registro. Una vez que llenamos el arreglo con los datos como se muestra en la imagen, se procede a ejecutar la petición para llevar a cabo el registro al evento mediante el método create del archivo importado FormacionInDataService, este recibe como parámetro la variable data que se creó.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Por último, tenemos esta especie de menú de navegación para alternar entre las pantallas de registro, asistencia y validación, pero esto depende de la categoría del evento ya que existen diferentes categorías de eventos si el evento es de tipo externo aparecerá el botón de validación para eventos externos en dado caso que el evento no sea externo aparecerá el de asistencia.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Utilizando el método getEvent() ejecuta una petición http exportada del archivo exterior EventosDataService.js utilizando el método get par un evento especifico, recibe como parámetro la id del evento la cual se encuentra en la url de nuestra página. Obtenemos los datos de la categoría del evento y los guardamos en la variable categoría de la cual dependerá que botones se mostraran gracias a la directiva v-if.

Captura de pantalla de computadora

Descripción generada automáticamente

Los botones para navegar ejecutan un método llamado sendEvent() el cual utiliza el id del evento que se encuentra en la url para redireccionar a otra pantalla gracias al router, en este caso dependiendo de la categoría del evento te mandara a una pantalla u otra.

Captura de pantalla de computadora

Descripción generada automáticamente

## **Módulo de asistencia.**

Ubicado en la ruta pfi-front/src/components/FormacionIntegral/Asistencia.vue

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Este módulo cuenta con una interfaz parecida a la anterior solo que tenemos un botón adicional para aplicar una asistencia general a todos los alumnos registrados. Se utiliza un v-data-table, donde se muestran el nombre, matricula y tenemos la opción de eliminar alumnos del evento o poner una asistencia positiva o negativa. Este v-data-table recibe recibe como parámetros los headers para llevar a cabo los encabezados y tarjets, después recibe los datos de los alumnos registrados que están guardados en la variable alumnos.

Captura de pantalla de computadora

Descripción generada automáticamente

El arreglo de alumnos se obtiene al entrar en esta pantalla mediante el método retrieveAlumnos(), este ejecuta la petición http al archivo FormacionInDataService.js el cual retorna el método getAll() para obtener a todos los alumnos registrados a un evento simplemente pasando como parámetro el id del evento utilizado en las url de nuestra página.

Captura de pantalla de computadora

Descripción generada automáticamente

Dependiendo de los datos de la asistencia el color de los botones cambiara gracias a la directiva v-if, si la asistencia tiene un 1 cuenta como asistencia, si es un 0 cuenta como que no asistió y si es nula es porque aún no se ha puesto la asistencia. Estos botones ejecutan el método aplicarAsistencia() el cual recibe como parámetros el item o los datos del alumno y su valor numérico de asistencia 1 o 0.

Captura de pantalla de computadora

Descripción generada automáticamente

Al ejecutar el método mediante el clic del botón lo primero será tomar los datos tomados del alumno ya que es uno de los parámetros que se piden, se modifica en su valor de asistencia y después este llevara a cabo la petición mediante el archivo FormacionInDataService.js en su método update que recibe el id del registro de formación integral y los nuevos datos de los cuales solo cambiamos la asistencia y ya dependiendo del valor de la asistencia se mostrara un mensaje de lo sucedido.

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

A su vez el botón de asistencia grupal funciona mediante un ciclo for of el cual recorre el arreglo de alumnos y por cada iteración se ejecuta el método de aplicarAsistencia() pasando como parámetros al alumno y su asistencia.

Captura de pantalla de computadora

Descripción generada automáticamente

El botón de eliminar es para borrar del registro de alumnos al evento alguno que se haya agregado por equivocación, este funciona gracias al evento deleteAlumno() el cual recibe como parámetro el id del registro de formación integral. Este utiliza el archivo de FormacionInDataService.js para ejecutar la petición al back-end llamada delete la cual recibe el mismo parámetro del método deleteAlumno. Una vez que se ejecutó sin problemas se eliminara al alumno y se vuelve a ejecutar el método de retrieveAlumnos() para actualizar el contenido de la tabla además de mostrar un mensaje de los resultados.

Captura de pantalla de computadora

Descripción generada automáticamente

Una vez que se ejecutó sin problemas se eliminara al alumno y se vuelve a ejecutar el método de retrieveAlumnos() para actualizar el contenido de la tabla además de mostrar un mensaje de los resultados.

Interfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamente

## **Módulo de validación de créditos.**

Se ubica en pfi-front/src/components/FormacionIntegral/Validacion.vue

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Este módulo funciona prácticamente como el de asistencia su única diferencia es que este se utiliza para eventos externos los cuales requieren subir una evidencia para verificar que cumplieron con las condiciones del evento. Además, que los botones que anteriormente aplicaban la asistencia ahora son mera estética para mostrar si realmente se validaron los créditos, mientras que el botón de eliminar sigue igual. Entonces para acceder a la evidencia subida por el alumno debemos dar en su botón de verificar evidencia, esta desplegara una ventana modal la cual muestra la imagen, así como tres botones que son para validar los créditos, no validarlos y cerrar la ventana modal, pero lo importante es que además de desplegar la ventana modal tambien ejecuta un método para obtener la evidencia, este método es retrievEvidencia().

Captura de pantalla de computadora

Descripción generada automáticamente

Este método recibe como parámetro la información del alumno cuando le damos clic al botón para ejecutar la petición http desde el archivo EventosDataService.js llamado getEvidencia(), este recibe el id del evento y el id del alumno el cual es una llave foránea en nuestros datos y está representado con el nombre alumno, por eso se indicó como data.alumno. Después se guardan los datos en la variable evidencia para poder utilizarl la imagen recibida, mientras que guardamos los datos del alumno en la variable datosValidacion ya que la ventana modal es un componente a parte del v-data-table, entonces no podemos acceder a los datos desde ese mismo componente.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Estos botones funcionan como los anteriores ejecutando el método validarCreditos() el cual es básicamente método de aplicarAsistencia() utilizado en el módulo de asistencia solo que con otro nombre para no ocasionar confusión y utilizando la variable datosValidacion para validar los créditos.

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

## **Módulo de alumnos para administradores.**

Se ubica en pfi-front/src/components/Alumnos/Alumnos.vue

Interfaz de usuario gráfica

Descripción generada automáticamente

Esta pantalla también utiliza un componente v-data-table el cual recibe como parámetros los headers establecidos.

Captura de pantalla de computadora

Descripción generada automáticamente

Este recibe los datos de los alumnos, se obtienen mediante el método retrieveAlumnos() el cual realiza una petición http para obtener los datos mediante el archivo AlumnosDataService.js con el método getAll(), este muestra a todos los alumnos registrados en el sistema y se guardaran en la variable alumnos que pasaremos al v-data-table. Para encontrar el alumno que se necesita solo es necesario usar el buscador.

Captura de pantalla de computadora

Descripción generada automáticamente

Al final del v-data-table hay un botón para ver más detalles el cual tiene el objetivo de ejecutar el método sendStudent(), este recibe como parámetro el id del alumno, ya que este redirige a una ruta dinámica para visualizar todos los eventos inscritos del alumno así como sus respectivos créditos y asistencias.

Captura de pantalla de computadora

Descripción generada automáticamente

## **Módulo de historial de eventos.**

Se ubica en la carpeta pfi-front/src/components/Alumnos/HistorialEventos.vue.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Este módulo cuenta con 3 apartados, el primero que muestra los datos del alumno, el cual obtiene sus datos mediante el método getAlumno() el cual hace una petición para recibir los datos del alumno gracias al id enviado desde la pantalla anterior, haciendo uso de este id podemos obtener los datos del alumno gracias al archivo de servicios AlumnosDataService.js el cual ejecuta el método get el cual recibe el id del alumno para retornar sus datos.

Captura de pantalla de computadora

Descripción generada automáticamente

Una vez que ofrece la respuesta se procede a guardar los datos en una variable llamada dataAlumno para mostrar los datos en el html, después se utiliza el método push para insertar en un arreglo alumnoDataFiles los datos del alumno en dado caso de querer exportar sus eventos a un pdf o Excel así como la variable fileName tiene el mismo objetivo obteniendo la matricula del alumno para usarlo como nombre del archivo, por último se guarda un texto para utilizar en dado caso de querer descargarlo en formato de Excel.

El segundo apartado es el que muestra todos los eventos a los cuales el alumno ha sido registrado, en este apartado funciona mediante la obtención de los eventos para mostrar los datos de estos y poder validar a cuáles se asistió o se validaron sus créditos dependiendo del evento. Estos datos se guardan en la variable eventsAlumno mediante el método getEventsAlumno() la cual hace una peticion al archivo FormacionInDataService.js encargado de los registros de eventos y ya teniendo la respuesta guardamos los eventos y ejecutamos un segundo método para validar los créditos llamado validarCreditos() al cual le pasamos la respuesta o directamente la variable eventsAlumno.

Captura de pantalla de computadora

Descripción generada automáticamente

Lo primero que hace es en definir el valor de la variable eventsDataFiles la cual es utilizada para exportar los evento a formato Pdf o Excel la cual es utilizada en el método createData(), despues continuamos con un ciclo for of para recorrer todos los eventos e ir sumando los puntos de los créditos dependiendo si la asistencia o validación del evento es positiva con el uso de un valor numérico 1 para guardar los datos en la variable totalCreditos y al final ejecutar el evento createData() el cual recibe la variable evento la cual lleva los datos del evento por cada iteración del ciclo.

Captura de pantalla de computadora

Descripción generada automáticamente

Así que por cada iteración se cambia el atributo asistencia del objeto, remplazando el valor numérico por un texto el cual dependerá de la variable status la cual se declara dependiendo del valor numérico de la asistencia, ya sea 1, 0 o nulo para el final utilizar el método push de un arreglo para insertarlo y guardar sus datos, con la finalidad de poder exportar los datos de los eventos a un Excel o Pdf.

Captura de pantalla de computadora

Descripción generada automáticamente

En el tercer apartado encontramos lo que son las funcionalidades para exportar los eventos a un Pdf y Excel las cual a lo largo de los métodos que se han visto se han obtenido los datos necesarios para crear estos documentos.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

El botón con el icono de pdf tiene la funcionalidad de ejecutar de ejecutar el método createPdf() el cual funciona de la misma forma que el pdf creado en el módulo de eventos. La diferencia se encuentra en que esta muestra primero al alumno en lugar del evento y después los eventos en lugar de los alumnos. Por lo que en la primera tabla definimos las columnas con su tarjet y su títulos para pasar en el body los datos a mostrar que serían los que se crearon en el método getAlumno() donde guardamos los datos en la variable alumnoDataFiles.

Captura de pantalla de computadora

Descripción generada automáticamente

Mientras que en la segunda tabla definimos las columnas con su texto y respectivo tarjet para al final pasar en el body la variable eventsDataFiles utilizada en el método createData().

Captura de pantalla de computadora

Descripción generada automáticamente

Al final solo añadimos el pie de página y guardamos el archivo con el método sabe agregando el nombre del archivo y su extensión. Para mostrar el resultado final.

Sitio web, Escala de tiempo

Descripción generada automáticamente

Por último, tenemos el Excel el cual se elabora con el uso de la librería vue-json-excel, entonces hay que importar los archivos necesarios en el script y añadirlo a los componentes.

Captura de pantalla de computadora

Descripción generada automáticamente Captura de pantalla de computadora

Descripción generada automáticamente

Solo hay que insertarlo en nuestro html, este componente funciona como un botón por lo que al darle clic se ejecutara su funcionamiento y solo hay que pasarle mediante los props los datos que necesita.

Captura de pantalla de computadora

Descripción generada automáticamente

Solo hay que pasar la data que ya creamos, este son los datos guardados en la variable eventsDataFiles, se le pasa el nombre y su extensión, el header sería la primera fila del Excel la cual puede ser un título o algo importante, en este caso son los datos del alumno, mientras que el footer es la última fila del Excel en la cual es el lugar para los créditos válidos. Por último, tenemos los fields que son las columnas con su texto y el valor del tarjet como se muestra a continuación.

Captura de pantalla de computadora

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

## **Moduló de calendario.**

Se ubica en la ruta pfi-front/src/components/Eventos/CalendarioEventos.vue

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Este módulo tiene 3 apartados el primero es un componente de vuetify llamado v-calendar el cual recibe como parámetros la variable events. Para conocer la estructura de este componente visite la documentación para más información ya que solo se muestran los métodos que no son de este.

Captura de pantalla de computadora

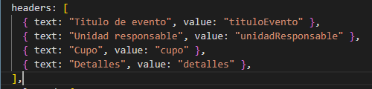
Descripción generada automáticamente

Esta variable obtiene sus datos por el método getEvents() el cual hace una petición http a través del archivo EventosDataService.js el cual retorna los datos de todos los eventos registrados en el modelo de calendario mediante su método getAllCalendario(), ya con una respuesta positiva almacenamos los datos en la variable events con la estructura necesaria diseñada desde el modelo de la base de datos y es la que usaremos en el componente v-calendar.

Captura de pantalla de computadora

Descripción generada automáticamente

El segundo recuadro solo muestra los eventos que existen el día de hoy, este recuadro es un componente v-data-table el cual recibe los eventos del día mediante los props de ítems y sus headers.



La variable events obtiene sus datos del método getEventsToday() la cual ejecuta la petición http para retornar los eventos del día mediante el método getTodayEvents del archivo EventosDataService.js el cual recibe como parámetro un día, en este caso hoy el cual se obtiene gracias al evento getDate() para obtener la fecha del día y retornar la fecha.

Captura de pantalla de computadora

Descripción generada automáticamente

Estos dos apartados tienen la opción de mostrar mas información acerca del evento, mediante el componente infoEvento.vue utilizado anteriormente el cual recibe como props los datos del evento, estos se obtiene mediante el método sendEvent(), este método recibe el id del evento para ejecuta la petición y obtener los datos para guardarlos en la variable selected la cual es la que recibirá el componente.

Captura de pantalla de computadora

Descripción generada automáticamente

## **Módulo de vista de alumnos.**

Se ubica en la ruta pfi-front/src/components/AlumnosVista/EventosAlumno.vue

Interfaz de usuario gráfica

Descripción generada automáticamente

Ahora pasando con la parte de la Interfaz del Usuario, esta parte es lo que se le mostrar al alumno una vez que entra en la sección de sus eventos registrados, en donde se puede observar en la parte izquierda una tabla en donde el alumno podrá encontrar y buscar sus eventos registrados y del lado derecho otro cuadro en donde se mostrará la información del evento que seleccione el alumno.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Cuando el alumno seleccione un evento utilizando las checkbox es cuando se mostrará la información correcta de cada evento, dependiendo de la selección, donde se mostrará gran información del evento.

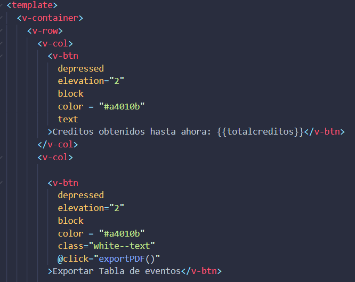


En la parte superior de ambas tablas se muestran dos secciones, una de ellas indica los créditos que el alumno ha obtenido de acuerdo con sus eventos registrados (parte izquierda) y en el otro lado se encuentra en botón de exportación para el alumno, en donde el alumno puede exportar la tabla que se le muestra, a un archivo PDF si así lo requiere.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En el caso de cuando se encuentre en un evento de categoría externo, la interfaz dará acceso al alumno a un botón con el cual lo guiara a otra sección de la plataforma donde el alumno podrá subir sus evidencias de dicho evento.



El módulo comienza con una programación muy común en el que se definen dos botones que estarán en la parte de arriba de la ventana que verán los alumnos, una definida para los créditos que el alumno obtiene y el otro que le permitirá al alumno acceder a la función de exportación por PDF.

Texto

Descripción generada automáticamente

A continuación de la codificación anterior, se define las características y propiedades que tendrá la tabla en donde se le mostraran al alumno los eventos registrados, haciendo uso de las funciones del Framework Vuetify.

Texto

Descripción generada automáticamente

A continuación, se importan los comandos necesarios para poder consumir la información que se obtiene de la base de datos desde la codificación y además también se importan librerías necesarias para el funcionamiento de exportación, y por último se incluye una tabla que se encuentra creada en otra carpeta del proyecto, y debido a que se va a utilizar, es necesario que se haga la exportación de ella.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

A continuación, utilizando las funciones de JavaScript se comienza por generar las diferentes variables que se utilizaran y una matriz en donde se almacenara la información necesaria de cada evento, además también se determina que objetos se mostraran en la tabla que se creó anteriormente.

Captura de pantalla con letras blancas

Descripción generada automáticamente

Empezando por el funcionamiento de los métodos que ayudan a filtrar los eventos que cada alumno tiene registrados, el primer método filtra los eventos que están registrados por la matricula del alumno y se almacena en una variable llamada “formacionesbymatricula”.

Texto

Descripción generada automáticamente

Una vez que los eventos están almacenados en la variable ahora se filtra un nuevo Array en donde se almacenaran eventos a los que el alumno tenga una asistencia positiva o que aún no se ha registrado asistencia ni negativa ni positiva.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Por último, para terminar con este método ya solo falta almacenar ambos Array en los que ya se definieron globalmente anteriormente, para poder utilizarlos en los métodos siguientes.

Texto

Descripción generada automáticamente

En el siguiente método se pide los eventos de acuerdo con los filtros obtenidos anteriormente, donde primero se comienza por pasar la información de los Array generales a unos locales de método, para no tener problemas al momento de usarlos dentro de este.

Captura de pantalla de un celular

Descripción generada automáticamente

En esta parte del código se aplica la función “map” a ambos Array para poder sacar solamente la información de los “ID” de los eventos filtrados y se almacenan en la variable “ids” e “idscreditos”, una vez que ya se obtuvieron las ids de los diferentes eventos, ahora si se obtiene toda la información necesaria de cada evento que coincida con la ip filtrada, creando un array con la información completa de cada evento en este caso variable “eventosbyid” y en la variable numero dos se almacena solamente los créditos de los eventos filtrados con asistencia positiva.

Texto

Descripción generada automáticamente

En la siguiente parte del código se muestra el método en el que se calcula la suma total de los créditos que obtiene el alumno de sus eventos a los que asistió, haciendo uso de la función “map” para solo obtener los datos de la sección créditos de cada evento, y a partir de este Array ya solo se suman los datos que se encuentran en él y pasan a almacenarse en la variable general. Y por último ya solo se pasa el array generado de la información de cada evento filtrado a la variable general y la cual es utilizada por la tabla para mostrarse en la interfaz

Texto

Descripción generada automáticamente

El ultimo método que utiliza este módulo es para hacer la exportación de la tabla de eventos que tiene el alumno registrado, en este caso la función hace generar una tabla con la misma información que se encuentra en la tabla virtual del alumno, y la función pasa esta tabla a un archivo tipo PDF, y se define las características de este documento, junto con el nombre que se guardara este archivo.

Texto

Descripción generada automáticamente

Por último, ya solo se pide que los métodos se inicialicen cada vez que se carga esta sección de la plataforma, para aplicar toda la función necesaria del funcionamiento que debe de haber normalmente.

## **Moduló de evidencias.**

Se ubica en la ruta pfi-front/src/components/AlumnosVista/EvidenciasAlumno.vue

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Para acceder a este módulo es necesario que el alumno este registrado a un evento externo, inmediatamente el componente EventoInfoAlumno.vue importado en el apartado de components para su utilización detecta el tipo de evento recibiendo como prop los datos del evento al seleccionarlo y dependiendo de esto ofrece la opción de subir una evidencia o actualizar, para esto envía el id del evento por una url dinámica.

Captura de pantalla de computadora

Descripción generada automáticamente

Este módulo cuenta con 2 apartados que funcionan de la misma forma y mediante la directiva v-if se verificará si ya existe una evidencia subida, dependiendo si hay o no hay te dará la opción para subir o actualizar evidencias.

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Este módulo muestra un input para seleccionar un archivo de tipo imagen, un botón para subir actualizar y a la derecha la evidencia y en dado caso de no tener aparecerá una imagen por default.

Captura de pantalla de computadora

Descripción generada automáticamente

Primero se utiliza el método retrieveEvidencia() el cual recibe como parámetro el id del evento y el id del alumno, estos serán utilizados en la petición http con el método getEvidencias desde el archivo EventosDataService.getEvidencias() y dependiendo de la longitud de la respuesta se decide que mostrar en pantalla, además de guardar los datos en la variable evidencia.

Captura de pantalla de computadora

Descripción generada automáticamente

Vue tiene una directiva llamada @change la cual ejecuta un método una vez que el componente tiene un cambio, en este caso cada vez que el input cambie obtendremos los datos de la imagen seleccionada pasando como datos la información del input utilizando $event en el método getImage().

Captura de pantalla de computadora

Descripción generada automáticamente

Cada vez que haya un cambio en el input se ejecutara este método el cual recibe sus datos y dentro de un arrego de objetos guarda la información necesaria para hacer la petición debido a que se utiliza un form-data para enviar los datos. La variable active está vinculada al botón para evitar que se realicen peticiones si no se seleccionado ninguna imagen.

Captura de pantalla de computadora

Descripción generada automáticamente

Es importante que el input se encuentre dentro de un formulario en el cual ejecutaremos la directiva @submit.prevent para tomar los datos de un submit y evitar que la pagina se recargue pasando un método el cual se ejecuta para realizar la petición, en este caso el método créate y el método update, los cuales funcionan de la misma forma con la diferencia que una utiliza el método post y el otro el método put.

Captura de pantalla de computadora

Descripción generada automáticamente

El evento necesita primero inicializar una variable FormData() para mandar los datos y el archivo por medio de una petición, utilizando esta misma variable con el método append el cual recibe como parámetros el nombre del campo, seguido de la información para cada dato necesario como lo es la imagen, el id del evento y el id del alumno.

Una vez que se guarden los datos en el formData se ejecuta la petición del archivo EventosDataService.js en su método createEvidencia el cual recibe como parámetro la variable formData.

Captura de pantalla de computadora

Descripción generada automáticamente

La función update funciona de la misma forma, solo que aquí la petición recibe como parámetro la variable formData y después el id de la evidencia ubicada en los datos guardados en la variable evidencia.