Documentacion proyecto ISI, frontend:

(Nota: Se recomienda trabajar el proyecto utilizando el editor de texto visual studio code o similares)  
Al entrar a la carpeta donde se encuentra el proyecto lo primero que vemos es una carpeta llamada “frontend” .Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En esta carpeta podemos encontrar los diferentes directorios y archivos del proyecto:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

-Los últimos 3 archivos son archivos que vienen por default ya sea en javascript o en la instalación de vite, preferiblemente se recomienda no modificarlos ya que son los que hacen que el proyecto funcione.

-El archivo readme.md es un archivo de texto vacio, no toma importancia en el proyecto.

\*node\_modules :

En la carpeta node\_modules podemos encontrar una gran variedad de carpetas y archivos dentros que vienen por default en la instalación de node, no podría dar una explicación de que hace cada uno pero puedo decir que es indispensable para que el proyecto funcione y se recomienda encarecidamente no modificarlos:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteCaptura de pantalla de un celular

Descripción generada automáticamente

En resumidas cuentas podemos encontrar las instalaciones de vue.js+vite además de las funciones básicas de node.

A continuación dejare videos de como instalar Node.Js, Vue.js y Vite en caso de que el proyecto no cuente con esta carpeta:

<https://www.youtube.com/watch?v=gG7E-n2fjmU>  
<https://www.youtube.com/watch?v=6mZRo4WR-bw> (Nota: UNICAMENTE hacer la instalación, la creación de carpeta se realizara con Vite en el video siguiente)

https://www.youtube.com/shorts/-C-w07\_LoP0

Una vez nuestras instalaciones estén hechas, debemos entrar a nuestra carpeta del proyecto desde la terminal con los siguientes comandos:

1. cd frontend

2. npm run dev

Esperas unos segundos y después de un rato podrías abrir y trabajar en el proyecto como es debido.

\*public :

En la carpeta public tan solo encontraremos 2 archivos, uno de ellos es el icono que utiliza nuestro proyecto en el navegador al abrir la pagina y el otro es un .svg que viene por default con vite a la hora de hacer un proyecto:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

\*src :

En src es donde esta lo principal que trabajamos durante este proyecto y encontraremos las siguientes 4 carpetas:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

\*src/assets :

Dentro de la carpeta de assets encontraremos 2 archivos:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Uno de ellos es el logo que utilizamos para el header de la pagina y que hace funcionar bien la apariencia del mismo, sin embargo el otro es un .svg que viene por default en la instalación de vue:



\*src/components :

En esta carpeta encontraremos los componentes que se muestran al entrar a la pagina del proyecto los cuales iré explicando uno a uno dentro de esta documentación: Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

(Ayuda.vue, Elecciones.vue y MapaCurricular.vue no han sido implementados y practicamente son plantillas en blanco por lo que seran omitidos; por otro lado BocetoMapaCurricular.vue se encuentra en una fase temprana por lo que se explicara superficialmente, otro componente es Checkboxes.vue el cual fue utilizado unicamente a manera de testing)

\*src/components/NavBar.vue :

Lo primero que vemos es la parte del template donde va el htmlTexto

Descripción generada automáticamente

Al ser una barra de navegacion la tag que se utiliza es una de tipo nav, acto seguido estamos importando una imagen desde la carpeta assets para el logo que tendra esta navbar y nombrando al elemento como logo\_unison. Despues tenemos una etiquetea h2 condicionada que interactua con el componenete userMenu que sera explicado despues. Según si ese componente es llamado o no mostrara un texto distinto en la navbar, lo que se complementa dentro del div en el que si se llama el componente userMenu mostrara el nombre del alumno, se añadira un input de tipo checkbox al que en otro div se le dara una animacion, su grafico sera importado desde un sitio web y ya por ultimo al ser clickeado mostrara los datos del estudiante. Texto

Descripción generada automáticamente

La anterior imagen es el script que importa el componente UserMenu.vue y que al exportarlo utiliza la informacion del estudiante sacada del JSON; tenemos una clase data que hace que el checkbox este en un estado de desuso y despues un metodo con el que definimos que sucede cuando este esta en uso o desuso, ya por ultimo tenemos “emits” que cambia la informacion de la clase data a verdadera haciendo que el checkbox este en uso.

Texto

Descripción generada automáticamente

La anterior imagen es el apartado style del componente que es donde viene la parte de css del componente. “\*”= le da un margen y padding general a todo el componente a demas de una fuente y el color de la misma; “input[type = “checkbox”]” = le da opacidad, altura y anchura de 0 a la checkbox del UserMenu.vue; “.dropdown” = hace que la posicion de la check box al realizar click sea relativa para que de esta manera no afecte a la navbar; “.user-menu-container” = este le da estilo al contenedor del UserMenu que es donde aparece la informacion al hacer click en el checkbox, podemos ver que hace que la posicion sea absoluta para que esta siempre sea la misma, añade que su posicion debe de estar a un 40% de la derecha, le da un color de fondo el cual es un blanco, un padding de 0.4rem, un redondeado de los bordes de 0.5rem, le da una anchura de 35rem y una relacion proporcional entre la anchura y altura de 6/5; “.user-menu” = esto le da la apariencia a lo que sale dentro del user menu, tal como un color gris en la parte interna, un borde redondeado de .5rem, fuente de color negro , y un display de tipo grid con una template de columnas repeat(3,1fr); “.dropdown svg” = da formato al icono del checkbox haciendolo de color blanco, que cuando el cursor pasa por encima cambia el tipo de cursor a un pointer, una anchura de 2rem con una relacion de aspecto de 1 y la velocidad de la animacion.

Captura de pantalla de computadora

Descripción generada automáticamente

“.dropdown.clicked > svg” = cambia la posicion de icono del checkbox haciendo que cambie su direccion mirando hacia abajo ademas de añadir una animacion similar a la del estilo anterior; “.img” = este estilo es el del logo de la navbar al cual le esta dando una relacion de aspecto anchura-longitud de 1, una anchura de 4rem y un padding de 0.3rem; “.nav” = esto es referente a la misma barra y le da un display de tipo flex con una tonalidad azul (#004990), hace que los objetos sobre el mismo esten centrados, le da un padding que elimina los bordes que saldrian normalmente y un gap de 1rem que hace que los objetos no esten pegados unos con otros y con los bordes de la pagina; “h2” = este es referente al texto que se muestra como titulo en la navbar sea el acronimo de occe o el completo ademas de darle a la fuente un “peso” de 300; “h3” = aquí solo cambia el “peso” de la fuente a 300 y es una etiqueta similar a la anterior; “div.userMenu” = le da un margen hacia la izquierda automatico al userMenu, un margen a la derecha de 2rem, un display flex, alinea y justifica los elementos del userMenu al centro y le da un gap de 1 rem, “div.wrapper” = le da una posicion absoluta al wrapper y hace que el tamaño de la parte de arriba sea del 100% mientras que el de la derecha sea del 40%.

\*src/components/UserMenu.vue :

\*src/components/formLogin.vue :

Texto

Descripción generada automáticamente

Aqui tenemos la parte del html donde hacemos un section dentro de un div con la id de “formLogin” para hacer un cuadro en el que hacer un formulario para un login, tenemos una etiqueta h1 que le da un titulo a ese formulario. Dentro del formulario tenemos 1 div que le da las etiquetas a los input de email y password y otro div donde se encuentran esos input los cuales estan marcados como obligatorios fuera de ambos divs tenemos un button, este es un boton que reacciona al posicionar el raton en el con una animacion y al clickear en el.

Texto

Descripción generada automáticamente

Despues tenemos los estilos: “.data-entry” = aquí se le da un display de tipo flex a la entrada de datos centrando el contenido del mismo; “.labels,.inputs” = aquí se alinean de forma simetrica tanto las labels y los inputs creando una separacion entre si de 1rem; “#formLogin” = aquí hacemos el display de tipo flex y lo direccionamos en columnas para centrar el div junto a su contenido, ademas de darle una longitud de 80vh; “\*” = aquí solo definimos el tipo de fuente que utiliza; “form” = aquí hacemos nuevamente un display de tiplo flex direccionandolo por columnas para poder alinear correctamente el contenido del fomulario en el centro del section, ademas de darle una separacion de 1rem del mismo; “section” = aquí le damos forma al cuadro del sectiondandole un ancho de 600px, un margen de 0 arriba y abajo, y auto a los lados con un borde grueso de 4px de un tono gris y las puntas redondeadas a .6rem, tambien se le da una posicion relativa y un padding de 10 px;

Texto

Descripción generada automáticamente

“section::after” = una vez hecho el estelizado anterior era necesario hacer este para terminar de darle el diseño, en general lo que este estilizado hace es hacer que se vea una linea divisoria entre el titulo y el contendio del formulario, en su mayoria son cosas respecto al tamaño de la misma y su color pero tambien podemos ver como es que se le da una posicion absoluta para que nunca cambie y un z-indez de -1 para hacer que no toque los bordes del section; “h1” = aquí cambiamos el diseño del titulo dandole un color azul claro con un margen inferior de 60 px y un “peso” de 300; el estilizado “.campo” fue retirado del y reemplazado por “.data-entry”; “label” = aquí se cambia el color del texto de las etiquetas, se les da un “peso” de 400 y un margen inferior de 40 px para que haya separacion entre ambas; “input” = aquí hacemos que el texto que se escriba en los input sea negro, les damos un margen izquierdo de 20 px para separarlos de las etiquetas y uno inferior de 40px para seprarlos entre si, tambien se redondean los bordes y se define la altura/anchura, ademas de darle un padding de 5px y se le quita el contorno para poder darle un sombreado externo a cada input.

Captura de pantalla de computadora

Descripción generada automáticamente

Lo siguiente es “.btn” = aquí se le da formato al boton cambiandole el color del mismo, como del texto encima de el, se le da un padding de 10px arriba y abajo, y de 30px a la izquierda y derecha, ademas se le quita el borde visible y se rendondea. Tambien se cambia cuando pones el cursor encima del mismo y se le da animacion, ya por ultimo se le da un margen inferior de 20px para que no toque el section y un “peso” de fuente de 600; “.btn:active” = aquí el boton se desplaza hacia arriba ligeramente al clickearlo; “@media (min-width: 768px)” = lo siguiente es un media query en el que se cambian los margenes del titulo, etiquetas, input y el boton para que puedan ser vistos en cualquier dispositivo.

Texto

Descripción generada automáticamente

Por ultimo tenemos un script con 2 metodos que son los que le dan animacion al boton y un tercero el cual no fue implementado en su totalidad puesto que no te envia a una pagina del proyecto si no a una de repuesto.

\*src/components/DashBoard.vue :

\*src/components/Materias.vue :



El html de este componente podemos dividirlo en 2 partes, una de ellas es un titulo que debajo tiene 2 divs que dividen un texto explicativo de la simbologia respecto a las materiasTexto

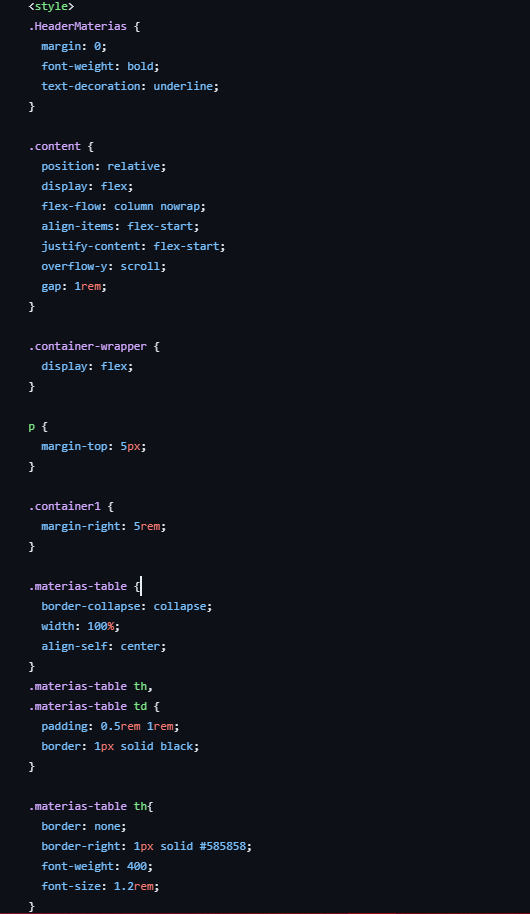
Descripción generada automáticamente

Y la otra parte es una tabla en la que hacemos una etiqueta colgroup y se crean diversas etiquetas col para hacer las columnas de la tabla, despues hacemos una etiqueta thead en la que dentro de una etiqueta tr le damos titulos a cada una de las columnas creadas empezando por “materias pendientes” y despues seguido de la simbologia ya explicada anteriormente. Ahora abrimos una etiqueta tr con un ciclo que llama al backend que se va encargar de hacer las filas de la tabla y llenarlas del contenido respectivo dependiendo del alumno.

Texto

Descripción generada automáticamente

Luego tenemos este pequeño script en el que nos sirve para crear una propiedad de tipo array para poder realizar la tabla anterior.



“.HeaderMaterias” = aquí se le da un margen de 0 al titulo, se cambia la fuente a tipo negrita y se subrraya el mismo, “.content” = aquí nos aseguramos de que todo el contenido tenga una posicion relativa con un display de tipo flex dandole un flujo de tipo “column nowrap”, tambien se alinea y justifica el contenido al inicio de la plantilla para que esta no este en en centro u otra parte, con overflow podemos seguir viendo el contenido fuera de la pestaña materias aun que estemos “scrolleando”, por ultimo se le da una separacion de 1rem a todo; “.container-wrapper” = aquí se le da un display de tipo flex a la cobertura del contenedor; “p” = aquí hacemos que cada parrafo del inicion tenga un margen superior de 5px para darles una separacion del titulo y entre si; “.container1” = aquí hacemos que esos mismos parrafos se encuentren todos ellos separados del borde derecho en 5rem empujando a su vez a los del “.container2” ; “.materias-table” = aquí empezamos a hacer cambios a la tabla colpasando el borde y alineandola a si misma al centro dandole ademas un ancho del 100%; “.materias-table th, .materias-table td” = aquí hacemos que tanto columnas como filas tengan un padding de 0.5rem arriba y abajo, y de 1rem a la izquierda y derecha dandole un borde de solido negro de 1px de anchura; “.materias-table th” = aquí llamamos nuevamente a la clase anterior para esta vez retirar los bordes de la parte superior de la tabla y cambiar el color para los bordes laterales de la misma parte. Tambien se cambia el “peso” de la fuente a 400 y su tamaño a 1.2rem;

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

“th.nombre materia” = aquí alineamos el texto para que aparezca de izquierda a derecha y no centrado; “.nombre-materia” = aquí hacemos un cambio a la anchura del area de esa columna en especifico; “.datos-materia” = hacemos lo mismo para el resto de columunas que el anterior solo que con menor anchura aun; “thead” = aquí se cambia el encabezado de la tabla para darle un color blanco y añadir un efecto de degradado de arriba hacia abajo en el que se vaya oscureciendo.

\*src/components/BocetoMapaCurricular.vue :

Texto

Descripción generada automáticamente

Lo primero que tenemos es el script en el cual tenemos un arreglo multidimensional donde cada variable tiene la informacion de cada materia del mapa curricular tal como los creditos, horas a la semana, requisitos, clave y nombre.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Lo siguiente es una clase que se encarga de generar cada uno de los cuadros del mapa curricular tanto en filas como culumnas dandoles formato y color con condicionales al inicio y despues haciendo divisiones dentro de cada cuadro para hacer la misma apariencia del mapa curricular. Por ultimo se manda a llamar la clase.

Texto

Descripción generada automáticamente

Lo que sigue es la parte del html donde tenemos una pequeña tabla que se encarga de explicar la simbologia de cada color.

Texto

Descripción generada automáticamenteY para terminar los estilos: “.container” = divide la pagina en un display grid para poder acomodar correctamente cada uno de los cuadros y los divide en 9 columnas a la vez que especifica el tamaño del patron en 70 px arriba y abajo, y auto a los lados. La siguientes funciones es alinear y justificar el contenido al inicio de la pagina y despues le da una separacion entre cada cuadrante de grid de 2px; da un tamaño total de 100vh a cada cuadro y le da un color blanco de fondo a toda la pagina y cuadros, tambien añade la fuente y su tamaño (16px); “.title” = le da al titulo de cada cuadro una fuente del tamaño del 50%, las hace negritas, le da un margen inferior de 2px y le permite al texto que en caso de que sea demasiado largo pueda continuar en la siguiente linea; “.subtitle” = le da un tamaño de fuente de 9px, le da un “peso” solido y un margen inferior de 1px; “.a” = se vuelve a definir el tipo de fuente para esta parte en especifico y su tamaño a 16px pero esta vez para la tabla de simbologia, se le da una anchira de 200 px con un borde solido de 3px y la posiciona al centro dandole un margen general de 20px con un display de tipo inline-block que permite definir la anchura y longitud dentro del mismo elemento; “.a td” = Alinea todo el texto de la tabla a la izquierda y da un padding de 8 px entre cuadros, y define los bordes laterales de la tabla; “.a tr, .a td” = crea los bordes interirores de la tabla con un pixel de ancho; “.aprobadas” = define el color de una tonalidad verde en caso de que la materia sea aprobada; “.reprobadas” = define el color de una tonalidad roja en caso de que la materia sea reprobada; “.proxima” = define el color de una tonalidad azul en caso de que la materia sea proxima.

El resultado de ambas es este: Imagen que contiene Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

\*services :

\*views :

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En views tenemos 3 carpetas y 4 archivos

-Los archivos: la funcionalidad de estos archivos es hacer que el proyecto pueda arrancar en el navegador, lo que sucede es que tenemos el app.vue que es lo que se muestra en el navegador

Texto

Descripción generada automáticamente

(En este caso es una plantilla en negro con la palabra “app”)

Y el index.html que ademas de decir el icono que se va mostrar en el navegador a llamar a app.vue

Captura de pantalla de un celular

Descripción generada automáticamente

En este caso quien enlaza ambos archivos es main.js

Imagen que contiene Texto

Descripción generada automáticamente

Importando los styles.css para que se puedan mostrar en el navegador

Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente

Estos vienen por defecto y son editables puesto que no hacen diferencia en nuestro proyecto por como esta construido, en general la funcion principal de este css es dar un formato y ubicación a la palabra app en el centro de la pantalla junto con el resto de la plantilla. La importancia de estos 4 archivos es funcionar como un dominio “home” para desde ahí acceder a las carpetas.

-Las carpetas:

Nota: la carpeta panel\_coordinador no esta empezada ni implementada por lo que es solo una plantilla en blanco e innaccesible, por lo mismo sera omitida

Nota 2: para ver los resultados se presupone que seguiste los pasos anteriores del principio del documento para poder ver los resultados de la pagina como si fueses un desarrollador.

\*views/login :

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En login tenemos 3 archivos similares a los vistos en views nuevamente, login.js enlaza de manera casi idéntica el index.html con la app.vue (LogIn.vue) a como vimos anteriormente

Interfaz de usuario gráfica

Descripción generada automáticamente

De la misma forma y con las mismas implicaciones ya vistas index.html manda a llamar a LogIn.vue



La diferencia radica en LogIn.vue

Texto

Descripción generada automáticamente

LogIn.vue importa desde componentes los componentes NavBar y formLogin y después los exporta dentro de la misma aplicación, siendo ese su script, mientras que en su template manda a llamar ambos componentes y en el caso de NavBar busca si ya se introdujo la información del usuario para hacer una diferenciación entre para saber que texto introducir en el titular de la navbar.

Para poder entrar a ver esta pagina como desarrollador se debe acceder a la siguiente ruta:

* localhost:5173/login/

este es el resultado: Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

\*views/panel\_estudiante :

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

panel\_estudiante.js:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Index.html:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Nuevamente la diferencia principal se encuentra en la app.vue que en este caso se llama “PanelEstudiante.vue”, a continuación este es el script:

Texto

Descripción generada automáticamente

En este caso se manda a importar los componentes de la carpeta components de NavBar.vue, UserMenu.vue, DashBoard.vue, Materias.vue, MapaCurricular.vue, Elecciones.vue, Ayuda.vue y los exporta dentro de la misma aplicación. También se manda a llamar “datosAlumno” y “materiasAlumno” desde la carpeta services para poder conectar correctamente el backend con el frontend con la clase data(). La clase data retorna una serie de valores, en este caso pide un numero de expediente el cual se da desde el código (de momento) y retorna los datos básicos del usuario y las materias que esta cursando en este momento. Con views se define que hace cada pestaña del componente DashBoard y manda a llamar sus respectivos componentes.

El script incluye 4 metodos: El primero hace que cambie lo que se muestra en la pagina según la pestaña y el componente seleccionado del DashBoard, el segundo carga los datos del usuario (alumno), el tercero carga los datos de las materias y el 4 obtiene esos datos formando asi con excepción del primer método un método constructor.

Texto

Descripción generada automáticamente

En la plantilla lo primero que hace es mandar a llamar el script de la app con el id de “panel”, después al igual que con login, verifica si se tiene los datos del alumno para cambiar el titulo y apariencia de la navbar con la diferencia que esta vez si lo hace. Luego se manda a llamar el componente DashBoard mientras llama las distintas clases y métodos que permiten cambiar la apariencia de la pagina según lo que elijas, por ultimo tenemos un componente en blanco cuya función es verificar en que pestaña se encuentra el DashBoard para asi mostrar determinada información lo que en este caso es las materias que esta cursando el alumno.

Texto

Descripción generada automáticamente

Por ultimo tenemos los estilos, “\*” = le dice al navegador que tenga en cuenta cualquier borde y relleno en los valores que especifique para el ancho y la altura de los elementos; “#panel” = le da una longitud al panel de 100vh, un display de tipo grid con una platilla de filas de 5rem X 1fr, y de columnas de 1fr X 7fr; “#dashboard” = le da un estilo con grid con una relación de tamaño de columnas 1/1 y de filas 2/3, además de hacer que los elementos alternem entre relativo y fijo, dependiendo de la posición de desplazamiento; “nav” = se le da una relación grid de tamaño a las columnas de 1/3; “.content” = aquí le da una relación grid de tamaño al contenido del panel de columnas 2/3 y de filas 2/3, además de darle un margen y un padding de 1rem; “::-webkit-scrollbar” = le da un estilo a la “scrollbar” de color transparente de fondo junto a una anchura de .5rem; “::-webkit-scrollbar-thumb” = aquí se le añade color a la “scrollbar” en el momento que la mueves haciendo que el color de la misma cambie a diferencia de cuando esta inactiva y además redonde a los bordes de la misma con un radio de .2rem; “.content \*” = hace que todo el texto del contenido sea negro.

Para poder entrar a ver esta pagina como desarrollador se debe acceder a la siguiente ruta:

* localhost:5173/panel\_estudiante/

este es un ejemplo del resultado de lo logrado:

Tabla

Descripción generada automáticamente