# /src/

Es donde esta el código principal

# /src/app.html

Es la plantilla básica del proyecto

# /static/

Los archivos estáticos

# /src/routes/

Donde van a estar las rutas del proyecto

Parece que la rutas se definen de forma automática a partir de este directorio, siendo este la raíz ‘/’, osea si se crea otro directoria seria la ruta /carpeta/

# /src/routes/+page.svelte

Este archivo que comienza con ‘+’ representa el código componente de la ruta, puede ser html

## +page.svelte

Va a estar dentro de cada carpeta ruta representando el código del componente

# /src/routes/+layout.svelte

Define el layout desde este directorio hacia abajo

## +layout.svelte

Se pone el contenido repetitivo y en el sitio donde se quiere que se renderize el +page actual se pone <slot/>

# [slut]

Para hacer una ruta dinámica, osea que parte de su contenido fijo/dinamico/fijo o /fijo/dinamico

Es con [nombre\_carpeta], en snake case

/fijo/[nombre\_carpeta]/+page.svelte

/fijo/[nombre\_carpeta]/fijo/+page.svelte

O puede ser parte del nombre de la carpeta

/fijo/[parte\_carpeta]x[otra\_parte]/+page.svelte

/fijo/[parte \_carpeta]x[otra\_parte]/fijo/+page.svelte

Estos sirven de parámetros

# /src/routes/+page.server.js

Este archivo que se puede ubicar en cada ruta, es para definir el comportamiento con el contenido de esta ruta se va a renderizar en el servidor

Por lo que veo es como que se define un contexto, mediante un dict o una lista y se le pasa a una vista como variables ya definidas

En una aplicación real en ves de importar los datos de data, serán pedidos desde una api

Data.js

export const post =[

{ slug: ‘Welcome’, title: ‘el titulo’},

{ slug: ‘Welcome2’, title: ‘el titulo2’},

]

+page.server.js

import {post} from ‘./data.js’;

export function load(){

return {

summaries: posts.map(

(post)=>{

slug: post.slug,

title: post.title

}

)

}

}

+page.svelte

Parce que recibe el resultado del load mediante una variable que hay que declarar llamada data

<script>

Export let data;

</script>

{#each data.Summaries as {slug,title}}

<li>{slug} – {title}</li>

{/each}

## Cookies

Si se va a trabajar con cookies se pueden utilizar desde el load

export function load( {cookies} ){

const variable=cookies.get(‘nombre cookies’);

cookies.set(‘key nueva cookie’, ‘nuevo valor’, { path: ‘/’ });

//codigo }

## POST

Para podere reaccionar a una llamada POST en esta ruta hay que agregar un actions a este archivo

page.server.js

Export const actions = {

default: async ({ cookies, request}) => {

const data = await request.formData();

let valorObtenido= data.get(‘nameinput’);

}

}

+page.svelte

<form method=”POST”>

<input name=‘nameinput’ />

…

</from>

### Multiples actions

Cuando hay mas de un action, no se puede poner el action default

page.server.js

Export const actions = {

nombreAccionA: async ({ cookies, request}) => {

const data = await request.formData();

let valorObtenido= data.get(‘nameinput’);

},

nombreAccionB: async ({ cookies, request}) => {

const data = await request.formData();

let valorObtenido= data.get(‘nameinput’);

}

}

+page.svelte

<form method=”POST” action=”?/nombreAccionB”>

<input name=‘nameinput’ />

…

</from>

No se limita solo a las acciones de este page, sino que tambien el form pude dirigir al de otra ruta

<form method=”POST” action=”/otra/ruta?/nombreAccionDeEsaRuta”>

# /src/routes/+layout.server.js

La manera anterior solo funciona para pasarle datos a un page, pero para que estos datos afecten a el layout correspondiente a la ruta tiene que ser con este archivo, el funcionamiento interno es igual

Estos datos son accesibles desde el propio page y layout de la ruta, pero tambien son accesibles para sus rutas hijas

# /src/lib/

Se puede ubicar archivos js que pueden se accedidos desde cualquier parte del código

/src/lib/micarpeta/miarchivo.js

Const dato=”dato”;

Un page o layout.server.js

Se carga usando ‘$lib/rutaAtu.js’

Import \* as mireferencia from ‘$lib/micarpeta/miarchivo.js’;

Const otroDato= mireferencia.dato

# Erroes

Se pueden lanzar errores con

throw new Error(‘descripcion’)