How does the Splat Mock work?

Raw Parsed

This is a summary of how the Splat Mock works, and its implementation details.

What is the Splat Mock?

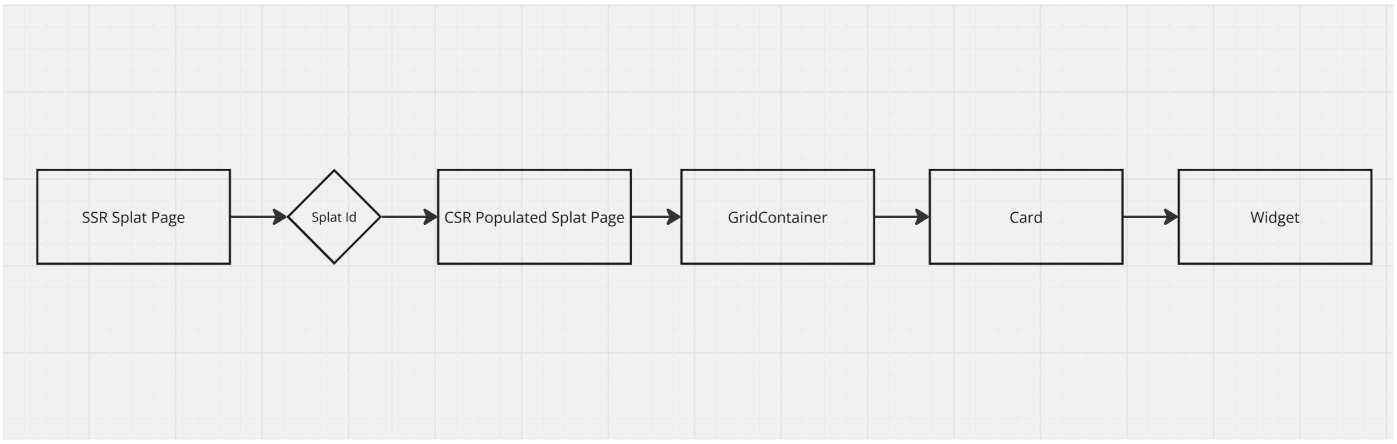
The Splat Mock is a 4x3 grid to mock of the Factsplat user interface. It is designed to be even more simple and goal oriented, as the dashboard only takes up one screen of space. The idea is to mimick most of the functionality of the original Factsplat interface, while experimenting with new features like placeholders to fit our applications needs. The splat is stored in a database as a JSON object, with the grid being a 2D array or matrix, and content having widgets being objects with their own properties. These objects can be looked at in the splat.ts file under the types/ folder. The grid types are under layout.ts in the same folder.

It has a few key features:

- As earlier mentioned, a 4x3 grid.
- Renamable title and subtitle at the top
- A toolbar to add new widgets
- Each widget on the grid can be resized as you wish in a grid fashion
- Each widget can be removed from the grid
- Each widget has placeholders that can be filled with data
- There are four types of widgets: Text widget, Icon widget, Member widget and Checklist widget.

How Does It Work Internally?

Let's explore the internal workings of the system from a left-to-right perspective, examining each component step by step.



Here's an image to help visualize the structure. Let's break it down:

Overview

1. **Server-Side Rendering**:
   - The journey starts with a server-side rendered page that uses a Supabase API endpoint to fetch a "Splat" by its ID. This logic resides in the `[id].tsx` file under the `/pages/splat` directory.

   - Once fetched, the Splat data is passed to the **Splat Page Component** for client-side rendering.

2. **Global State Management**:

   - On the client side, the Splat Page component populates the global state using Zustand. Zustand serves as the main mechanism for different components in the project to "communicate" with each other.
   - The Splat state includes the grid, content, title, subtitle, etc., all of which are managed in the `/store` folder.

Rendering the Grid

With the Splat state populated, the grid rendering process begins, as outlined in the diagram. The steps are as follows:

1. **GridContainer Component**:

   - This component handles the primary calculations for the grid. It is responsible for managing the layout and ensuring the grid logic is functional.

2. **Cards/Widgets**:

   - The grid content consists of cards or widgets, each packed with logic for resizing, moving, and updating the grid.
   - Widgets are rendered within the GridContainer and displayed to the user.

3. **Card Content**:

   - Each card can host children (in the React sense), making it straightforward to populate with any required content.
   - This design makes it simple to add new widgets—just create a new card with the desired content.
   - Additionally, all cards reuse the resizing logic and include a cross button for deletion.

Toolbar, Dialogs, and Additional Logic

- Beyond the grid, other aspects of the Splat, such as the toolbar or dialogs (popup elements), are used for adding new widgets, editing placeholders, and more.
- These components are located in the `/components` folder.
- Placeholder logic is implemented using a simple boolean flag in the JSON object of the widget content (referred to as a droplet).

Implementation Details

The Grid

The grid is a 2D array (or matrix) of IDs. For example:

Here we have a 4x3 grid with 8 widgets and one empty space.

The Content

Here is an example of the content object with only one widget, id: 1.

```
    content: [
      {
        id: 1,
        title: "Customer",
        type: "text",
        content: [
          {
            id: 1,
            title: "Sticos om Factsplat",
            url: "https://www.youtube.com/watch?v=yBpGR4K-1Xo",
            type: "link",
            placeholder: false,
          },
          {
            id: 2,
            title: "vatne@metaito.com",
            url: "mailto:vatne@metaito.com",
            type: "email",
            placeholder: false,
          },
          {
            id: 3,
            title: "+47 472 85 528",
            url: "",
            type: "phone",
            placeholder: false,
          },
          {
            id: 4,
            title: "SSID: Brygga",
            url: "K59@TrondheimNO",
            type: "text",
            placeholder: false,
          },
          {
            id: 0,
            title: "",
            url: "",
            type: "text",
            placeholder: true,
          },
        ],
      },
  ]
```

This is only an example of the text widget.

```
[
  [1, 2, 3, 3],
  [4, 5, 3, 3],
  [6, 7, 8, 0]
]
```