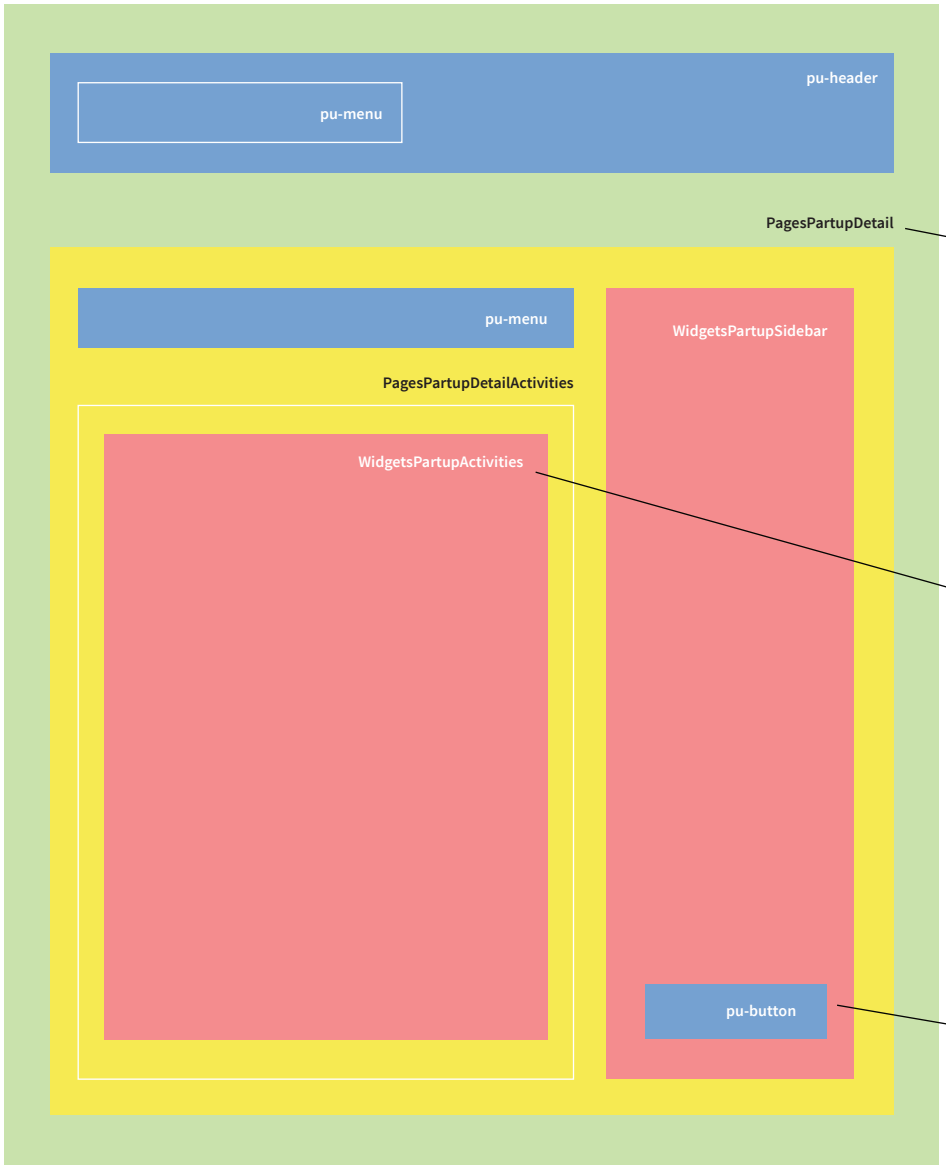
Frontend structure



Layout

Declared in: client/templates/layouts/app/
- LayoutsApp.html
- LayoutsApp.sass
- LayoutsApp.js

Layouts are the top-level templates. They can contain a header, current page placeholder and footer. The Sass file should only contain header and footer positioning rules. The js file should keep track of the state of the template and handle navigation functionality.

Page

Declared in: client/templates/pages/partup-detail/
- PagesPartupDetail.html
- PagesPartupDetail.sass
- PagesPartupDetail.js

Pages can contain single components with page-specific functionality, widgets (packages) and sub-pages. A page, in fact, only represents a composition. Therefore, the Sass file should only contain position defenitions of the inside components. The js file should handle the page states and navigation functionality if subpages are present.

Widget (package)

Declared in: packages/partup:client-widgets-partup-activities/
- package.js
- WidgetsPartupActivities.html
- WidgetsPartupActivities.sass
- WidgetsPartupActivities.js

With a funcionality, you can think of a widget which will fulfill one standalone functionality. Functionalities that tie the app together (like a navigation bar) should not be declared as a package, because it's not a widget with a standalone functionality. The Sass file may only contain component composition rules. When a widget is called *WidgetsPartupActivities*, the package should be called *partup:client-widgets-partup-activities*.

Small component

Declared in: client/stylesheets/components/
- pu-menu.sass

The whole app is made up of small styled components. These components are not functional by themselves, but only provides styling. For example: buttons, inputs, titles, paragraphs and menus. Each component should be defined as a Sass class prefixed with "pu-", for example "pu-button". Be aware not to define any styling dealing with the position of the component inside its parent or relative to its siblings.