Web Tech Report

Jergus Lejko *(jl17105)* and Rafael d'Arce *(rd17647)*

We have created a desktop-only web app called Busybee. Instructions on how to run it are available in the README. It is a Kanban board inspired task organisation tool. We recommend that you first try logging in with our demo user (see README) before creating your own. Visit the "About" page to learn more about functionality. We are proud to have created a responsive, easy-to-use app with a consistent art style.

HTML (A)

- PUG templating language to organise our templates.
- Modularised code (reusable components such as the nav bar).
- We used https://validator.w3.org/ to validate the generated HTML
- Initially inspired by W3 Schools tutorials for Modals (popups) and Navbars, but those components have since gone far beyond the tutorials.

CSS (A)

- We used various techniques such as floats and Flexbox to organise the layout
- jQuery UI for column and task portlets.
- We used CSS animations for popups/modals.
- Responsive buttons and cursor.

JS (A)

- jQuery UI for sortable/droppable functionality
- Modern fetch() API to communicate with the backend asynchronously after dragging and dropping.

PNG (A)

- Used GIMP to create an original artwork by handling layers with transparent backgrounds and using the airbrush tool to draw. Also imported and manipulated some shapes created on Inkscape as PNGs.

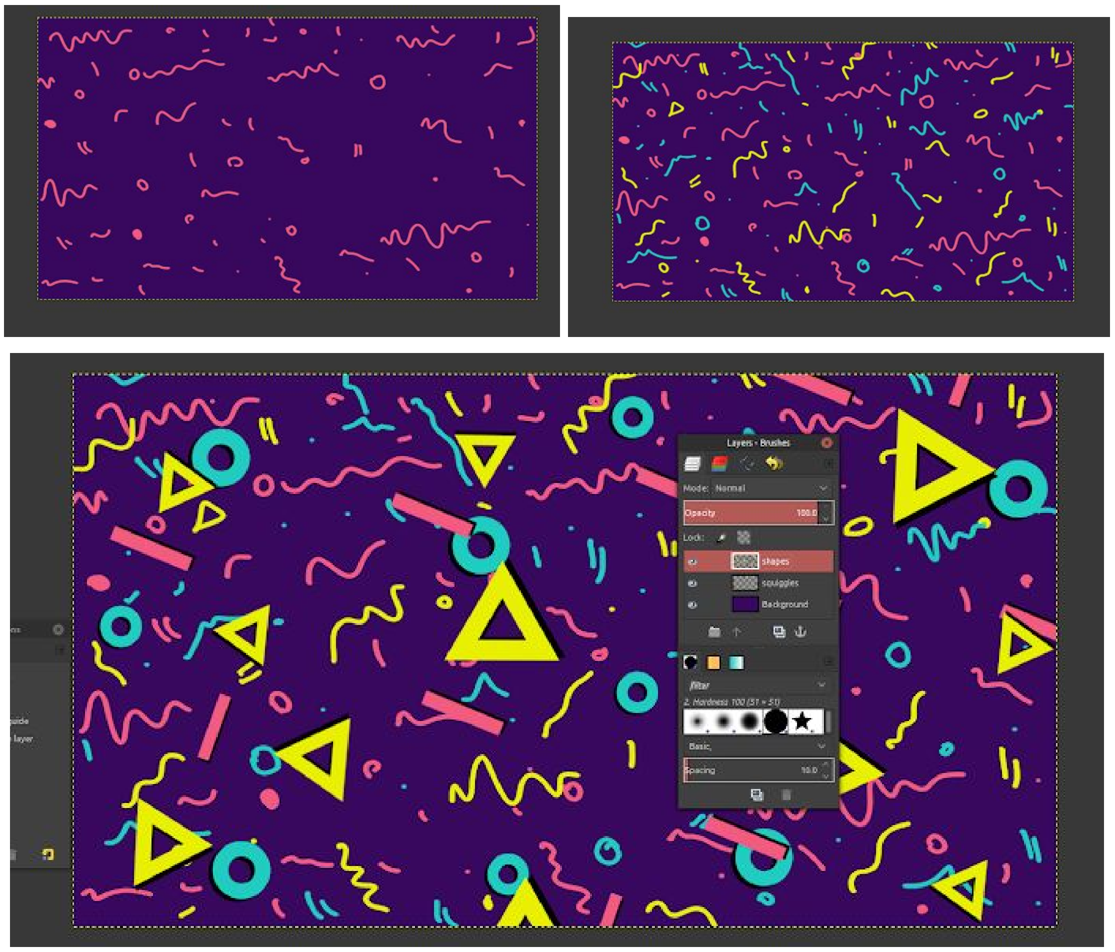

Figure 1: Creation of background image

SVG (A)

- Used Inkscape shape, colour, gradient, path editing (e.g. union, difference) tools to create SVG logo, button icons and shapes for background artwork.

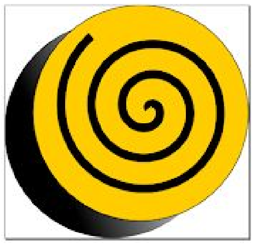
Figure 2: Logo using gradient

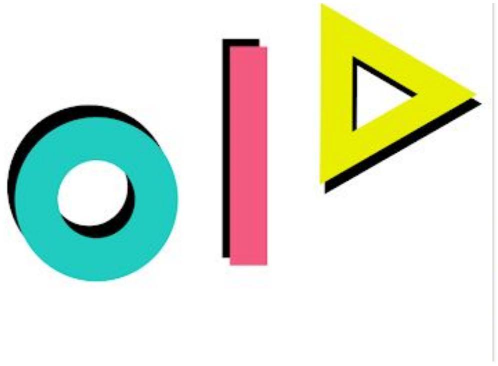
Figure 3: Use of path difference

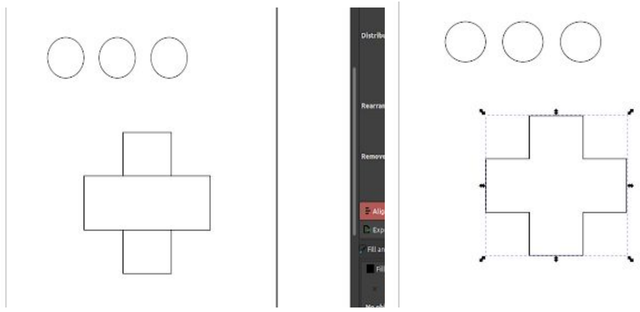
Figure 4: Use of path union for icon

Server (A)

- Express NodeJS framework to organise the web server
- Self-certificate HTTPS (because only ran locally)
- TypeScript to get type safety on the backend
- PassportJS library for authentication, as well as cookie-session library to get cookies working with ExpressJS
- Users can only access their own resources (boards, tasks) -- when accessing something they don’t own, they get redirected to the homepage
- We used Sequelize library as our ORM of choice
- Some pages have restricted access to them - they are behind a login wall. However, pages such as About can be access by anyone
- We validate the incoming data (login for example: username must be unique, password must be 6 or more chars)

CSS (A)

- SQLite3 database.
- We used Sequelize ORM to map tables to TS objects
- Randomised (using Faker) database “seed” script to populate the database with a demo user and data.

Dynamics Pages (A)

- Most of the pages in our app are dynamically generated such that they contain information specific to the logged in user -- users only see their boards, their tasks etc
- Responses are either HTML pages or HTTP redirects
- Pop-up forms are dynamically generated based on context (single modal template).

Depth (?)