Mateusz Jasiuk

JavaScript Developer

+48 692 083 522
mateusz.jasiuk@gmail.com
https://linkedin.com/in/mateuszjasiuk
https://github.com/mateuszjasiuk

EXECUTIVE SUMMARY:

I am a JavaScript developer with more than 6 years of professional experience. I'm currently working as a full-stack engineer with a focus on EventSourcing.

SKILLS:

- JavaScript
- TypeScript
- HTML/CSS
- Angular
- React
- Redux
- NestJS / NodeJS
- Event Sourcing
- Vim

LANGUAGES:

- polish - native
- english - fluent
- russian - conversational

ACADEMIC BACKGROUND:

BIAŁYSTOK
UNIVERSITY OF TECHNOLOGY

(B.E.), Computer Science
2010-2014

Bachelor's Work: Task organizer with a mobile client for Android devices.
Android / Java, PHP, HTML, CSS, JavaScript, MySQL, SQLite.

PROFESSIONAL HISTORY:

JAVASCRIPT DEVELOPER

Instapage and Postclick
December 2016 - Present

- Created parts of a front-end and a back-end for a collaboration tool for landing page builder using Angular2, NodeJS
- Created a feature for selecting and editing specific blocks in the page builder
- Moved main builder toolbar from top to sides, which included refactoring of most of the application dragging, resizing, measuring, snapping features
- Created company-wide feature for displaying information about the necessity of upgrading to the new plan. It included collaborating with every dev team in the office
- Created NodeJS microservice for blocking simultaneous users interacting with the page builder at the same time
- As a part of a two men team, developed an application that processed over 8 million pages and described the most likely data type that was supported by the page builder
- Went to the San Francisco office to work with the product team on a prototype of a new product that allows for creating landing pages at scale
- Created a PoC of the Automated Layouts backend which is now a base of a Postclick application. It included integrating NestJS CQRS plugin with EventStore
- Created parts of Postclick MVP like placeholders feature, block editor, style editor
- Introduced needed abstractions for backend like aggregate caching, aggregate cache invalidation, event versioning, persistent event bus

FRONT-END DEVELOPER

Astek / Roche
September 2015 - December 2016

- Created the front-end part of two applications using jQuery, HTML, CSS, React, Redux
- Bootstrapped and configured the second app using webpack2
- Convinced my tech lead to use React with Redux :)
- Worked remotely

WEB DEVELOPER

TangramCare
May 2014 - June 2015

- Created the front-end of applications using GWT, HTML, SASS, AngularJS
- Managed API of applications written in Spring
- Created original grid framework for internal use

PROJECTS SUMMARY:

TANGRAMCARE

WebScreeners: The application's goal was to improve the process of appointments in the doctor's office by allowing patients to fill pre-visit questionnaires as a replacement for doctor-patient interviews.
GWT, Java, Spring, CSS, HTML, JavaScript, PostgreSQL, MongoDB.

Discharge Director: Application that improves coordination between different medical institutions while discharging patients from hospitals.
AngularJS, SASS, HTML.

Annual Wellness Visit: The application that helps elderly people with their annual wellness visit. It's a multi-branched questionnaire that comprehensively checks all aspects of a patient's health. It has a module for the patient, physician, and clinician.
GWT, Java, Spring, CSS, HTML, JavaScript, PostgreSQL, MongoDB.

The Grid: The goal was to create a grid system that allows creating pixel-perfect layouts that are easily comparable to provided mocks. It also included dynamic font scaling, preserving the aspect ratio of containers, and zooming of the webpage up to 200%.
SASS, JavaScript.

ROCHE

iPipe: Portal for introducing and managing innovations. Roche has a lot of internal tools and a lot of employees. IPipe allows those people to share their ideas on how the company can improve in some areas. Every department can have its clone of iPipe if they want to introduce some ideas inside their structure.
JavaScript, jQuery, CSS, HTML.

Inventory: Spinoff of iPipie project, with inline editing and scaleable nature. Inventory is "iPipie light", with fewer features, but with an easier way to create clones for teams. It also improved performance a lot as it used React instead of jQuery as well as server-side rendering.
JavaScript, React, Redux, SASS, HTML.

INSTAPAGE AND POSTCLICK

Builder: Application for building landing pages. Instapage's Builder is a complex tool used for creating, publishing, and integrating landing pages.
It includes a lot of features like absolute positioning of widgets with draggable, resizing of widgets, snaping, measurements, adding blocks, adding popups, rich text editing, desktop, and mobile modes, A/B testing, custom scripts, inline style editing, and more!
JavaScript, jQuery, React, Redux, Angular, Ngrx.

Rick and Morty: There was a need for blocking two or more simultaneous users from editing the same page inside the Builder. Rick and Morty is a separate microservice that communicates with the Builder using sockets and adds every user that enters the page to the queue. When the first user in a queue lefts the page, then he/she has some time to return to it. If she won't then she is removed from the queue and another person can edit the page.
Angular, NodeJS, SocketIO, Redis, LUA.

Automated Layouts: As the Builder proved difficult to create personalized landing pages at scale, we created a new tool. Automated Layouts creates pages at scale, based on imported information from the client's ad network. The client would also specify his/her whole branding in the Style Editor which then would be applied to blocks of HTML created by us. Blocks themselves have placeholders for different types of content, like text, image, CTA, and more.
NestJS, Angular, Ngrx, HandlebarsJS, EventStore, Redis, MongoDB, fp-ts, io-ts, lerna.

Avengers: It's a preview module for the Builder. It includes the ability for real-time collaboration. Users can leave comments on the specific parts of a landing page so different experts like designers and copywriters can work together.
Angular, Ngrx, NodeJS, SocketIO, MongoDB.

JSON Bourne: Tool for analyzing and describing JSON for Builder pages. The Builder has some issues. One is that it has a lot of code, but no clearly defined space which defines its state(like redux store). Every page has its JSON file describing its internals, but it is "fluid" and changes often with time. Thanks to JSON Bourne we analyzed over 8 million currently(back then) published landing pages .json's. Because of that, we could create schema and typescript interfaces that describe those JSON files. We could also disregard fields that were a product of buggy behavior. This was the first step to introduce redux or some other state management into the Builder.
NodeJS, Typescript, Redis, JSON schema tools(e.g. quicktype.io).