qwikstubs

The Constructors | Alex Hamstra - Jared Roesch - Kyle Jorgensen - Ben McCurdy - Brittany Berin

THE PROBLEM

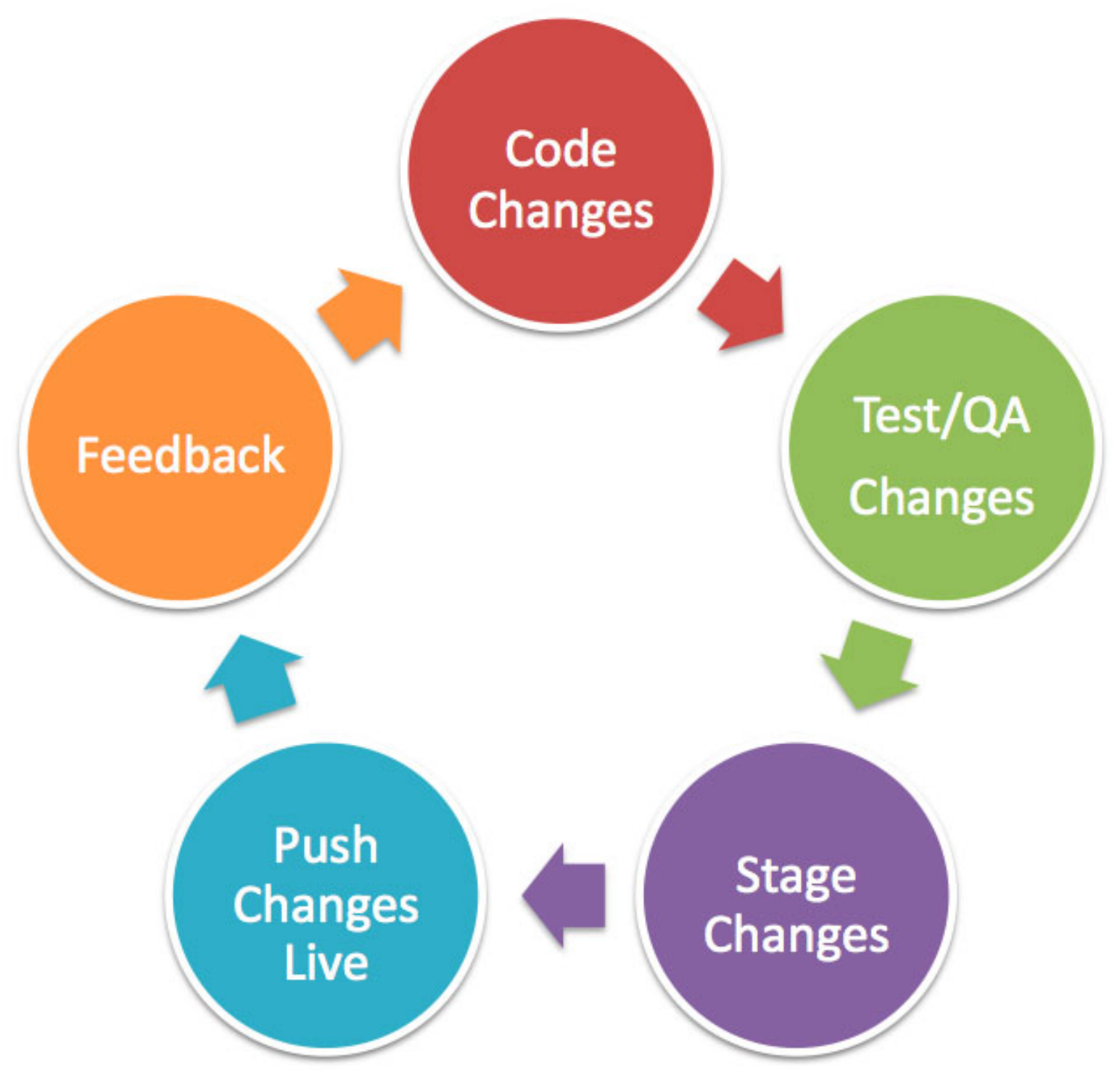
Today's online ticketing systems are inefficient and often create technical difficulties that make it nearly impossible for customers to buy the tickets they need. Seating arrangements can be unclear, their websites strain under high traffic, and tickets placed in a cart can be lost between page loads.

OPEN SOURCE

Our project is an open source endeavor. It is built from open source components, and is open to both contributions and inspection.

In this case open source doesn't mean second rate. We choose industry leading technologies to build our application.

Being part of the open source community was important to each team member, as well as our mentor.



TESTING

RSpec
A common testing framework for Ruby on Rails

Guard
Monitors our file system and automatically runs tests for specific files whenever they are changed

Travis CI
A continuous integration service that runs our tests whenever we push files to Github

THE SOLUTION

Qwikstubs combines the strengths of the technologies above into an unbeatable ticketing service. We combine the lightweight framework of backbone.js with the stability of Ruby on Rails to deliver fast page load times and a fluid user experience in a secure environment. MongoDB on the backend allows us to host huge events by handling thousands of requests per second without missing a beat.

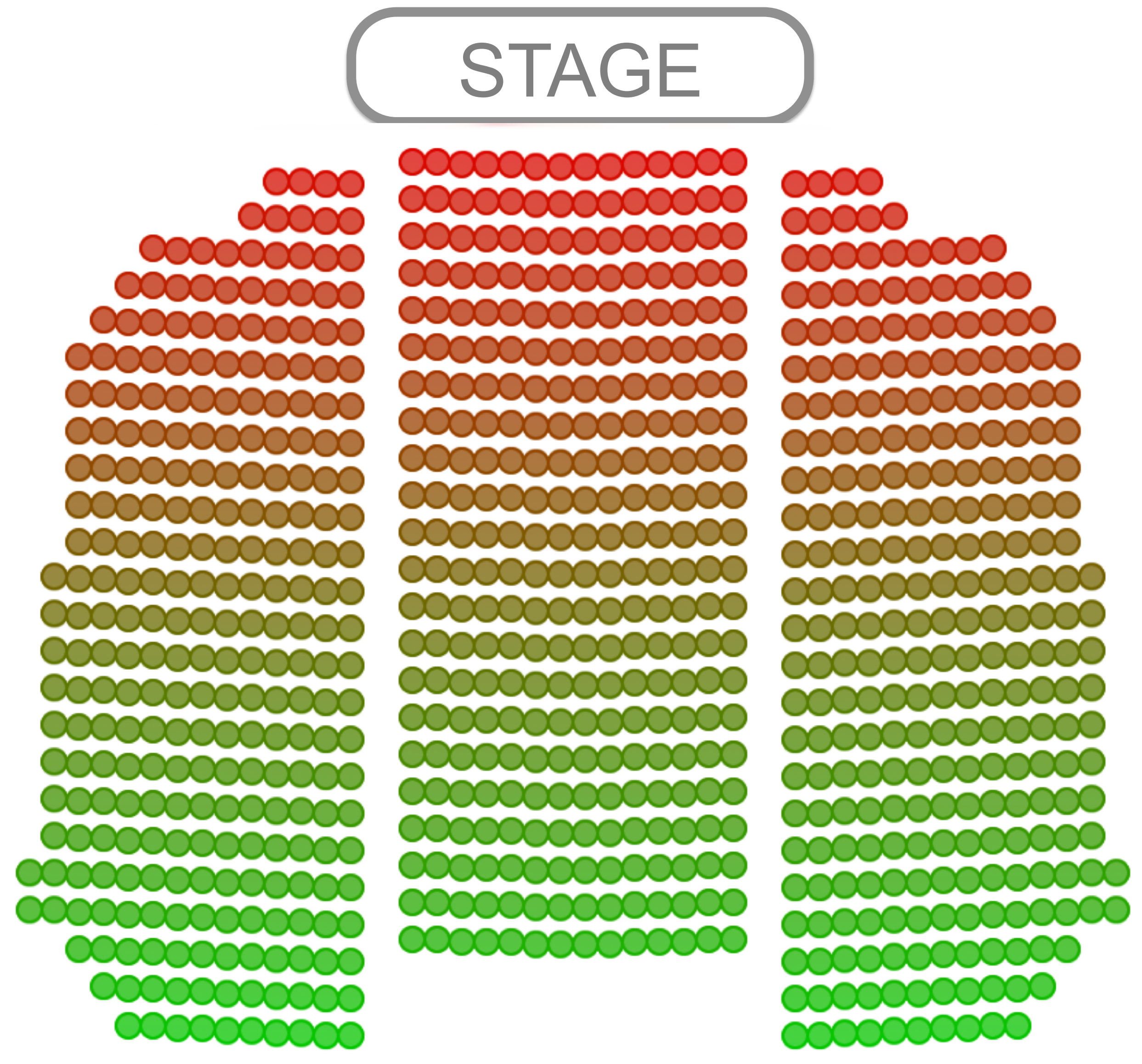
For the users who know what they are looking for, we have integrated Apache Solr with mongoDB to allow for quick pattern matching searches where they can find their event using a word or phrase in its title or description.

Additionally, we have the ability to browse upcoming events on a variety of devices with clean HTML/CSS styling via Twitter Bootstrap.

SEATING ALGORITHM

The algorithm is modeled after memory allocation, which uses a list for storing free pieces of memory. Our analogous solution to the free list is a collection that stores free chunks of seats, grouped by row. This process generates sets of the most desirable locations, as shown with the coloring in the seating chart below.

We can select unreserved seats with one query, allocate the requested amount, and return the rest to the free list. This allows the unallocated seats to be claimed by another party.



REAL TIME

We use Pusher, a modern publish-subscribe architecture to provide real time updates simultaneously to many clients. Each event is assigned a channel, that passes JSON objects in response to triggers. Consequently, as users reserve, purchase, or cancel their seats, their actions are instantly reflected in the seating view.

Acknowledgements | Mentor: Colin Kelley - Instructors: Chandra Krintz and Tim Sherwood - TA: Stratos Dimopoulos - Coaching: Janet Kayfetz