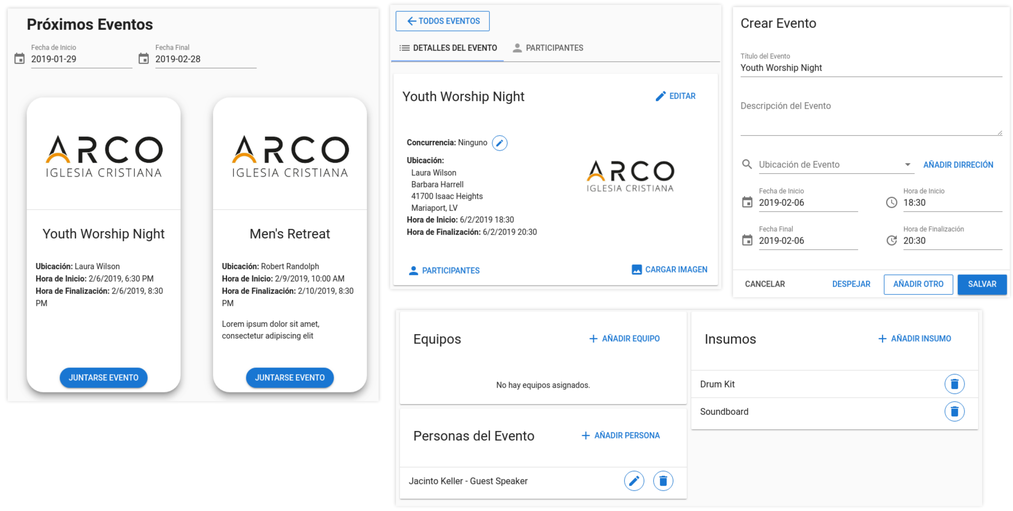
Corpus Christi: Event Planning

Jason Argo, Joey Ferguson, Matt Hapner, Noah Lindsey, Seth Lugibihl, Tim Ours, Grace Rose, Edric Yu

User Interface

We built a user-friendly interface to facilitate event creation and viewing. Admins can easily create and edit events, as well as add teams, people, assets, and participants to them. Admins can view events on the calendar or in a table form with quick action buttons. Public events can be seen in a card format where public users can self-register.



Our Experience

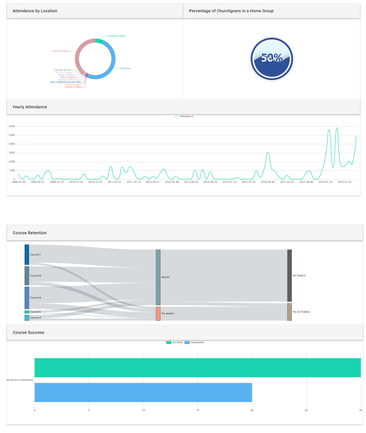
Key takeaways from the project:

- The value of horizontal collaboration, vertical collaboration, and open communication while working on a team or in an organization
- The vitality of flexibility in working in a different culture, on a new team, and in a new environment
- That as programmers, we have the immense capacity to serve the mission of the Church and a world in need

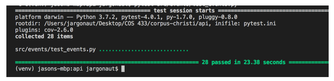
Through developing software for Arco and immersing ourselves in Ecuador's culture, our team gained a bigger perspective on the world and our vocation as computer scientists. We also had a lot of fun!

Dashboard

One goal of our application is to provide Arco with more than just a management hub for events; we want to provide them with analytics. By giving them the proper tools to analyze data, we hope to enable them to answer vital questions about the growth and longevity of their church. We created multiple visualizations - what we refer to as *dashboards*.

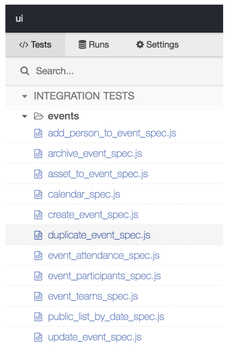


Quality Assurance



Back-End Testing

The primary tool we used was Pytest which ran functions in the API to check if they were executing as expected. We developed our tests in a way that guaranteed they would pass even as more functionality was added. Testing on the back-end was done to both API and database code.



End-to-End Testing

We did our end-to-end testing with Cypress. This consisted of mocking a real user, going into the web app, and performing particular tasks. It also enabled us to discover bugs and errors in the UI that could be quickly fixed, while also ensuring quality before merging code into production.

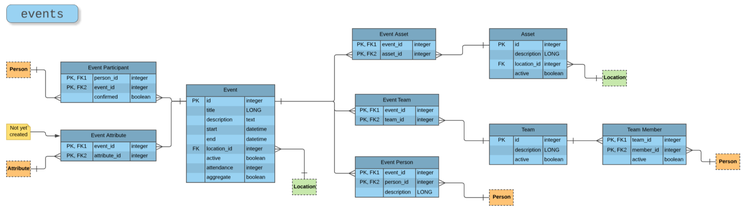
Conclusions

We were able to build out the core of the system's event planning functionality. Users are able to create, update, duplicate, and archive events and their corresponding teams and assets. We also built the skeleton for dashboard features.

Future Work

In the future, we look forward to adding image functionality, adding email notifications, simplifying a few parts of the UI, and deploying a first version of the system to production.

Back-end / API



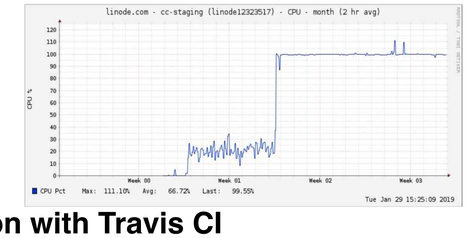
The back-end of our application consisted of a SQLite database for development and a PostgreSQL database for production. We implemented a RESTful API with Flask and Python to provide data back and forth between the client and database. The database tracks events as well as the event's participants, persons, teams, and assets. Additionally, images can be uploaded to the server's file system and added to events.

Dev-Ops

We purchased the domain name *corpus-christi.church* and SSL certified it using *Let's Encrypt*.

Server Dependencies

- Ubuntu 18
- PostgreSQL
- Nginx
- Supervisor



Continuous Integration with Travis CI

For every push to the Git repository, Travis CI runs a specific number of tests. First, the server environment is examined. Then the tests built by our own Quality Assurance team are run. If all tests pass, then the new code is pushed in real time to the server. This enables the user to have the most up to date version of the website.

Acknowledgements

We would like to thank our fearless supervisor, Ben Roller. We also would like to thank Dr. Ken and Greta Kiers for their continued support. Our final thanks to Dr. Tom and Darci Nurkkala for their expertise, leadership, and supreme logistical contributions to this project.

Tools & Technologies

SQLite, PostgreSQL, Flask, Vue.js, Vuetify, POSTMAN, SQLAlchemy, pytest, cypress, GitHub, Travis CI