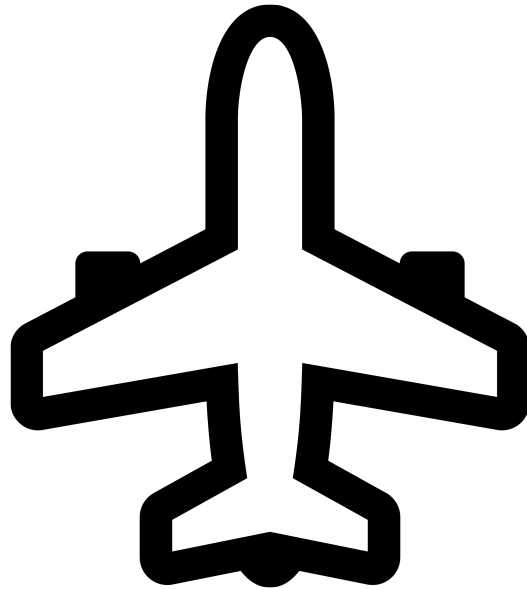# Integrating Drupal
with 3rd party API's

brought to you by
Nick Tubbs and Bill Juda

Travel Registry

Purpose

Capture the travel itineraries of Cornell Students and Faculty traveling abroad in order to notify them via text and email in the event of an emergency in their destination location.

Emergency Notification Needed

Concur Locate

Only needs small amount of info
- Traveler Name
- Dates
- Location
- Email
- Phone

Beyond Emergency Notification Data

We want to collect more information
- ITART
- Emergency Contacts
- Reason

Can't be done in Concur Locate
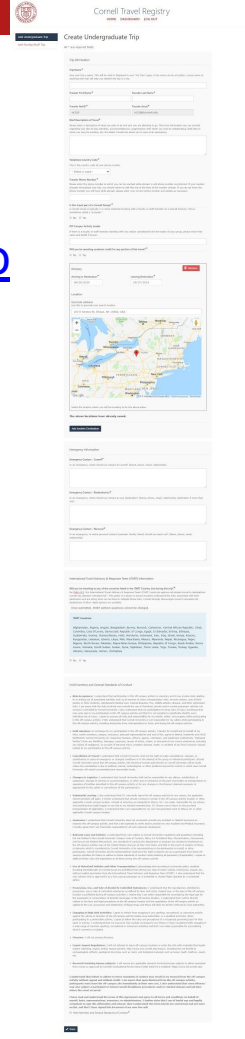
Control of Presentation

- Drupal can collect and connect the necessary data for the traveler.

- We can control the presentation of this form and data when we build it in Drupal

Architecture in Drupal

- 3 Content Type
  - Graduate Trip
  - Undergraduate Trip
  - Faculty/Staff Trip

- Paragraphs
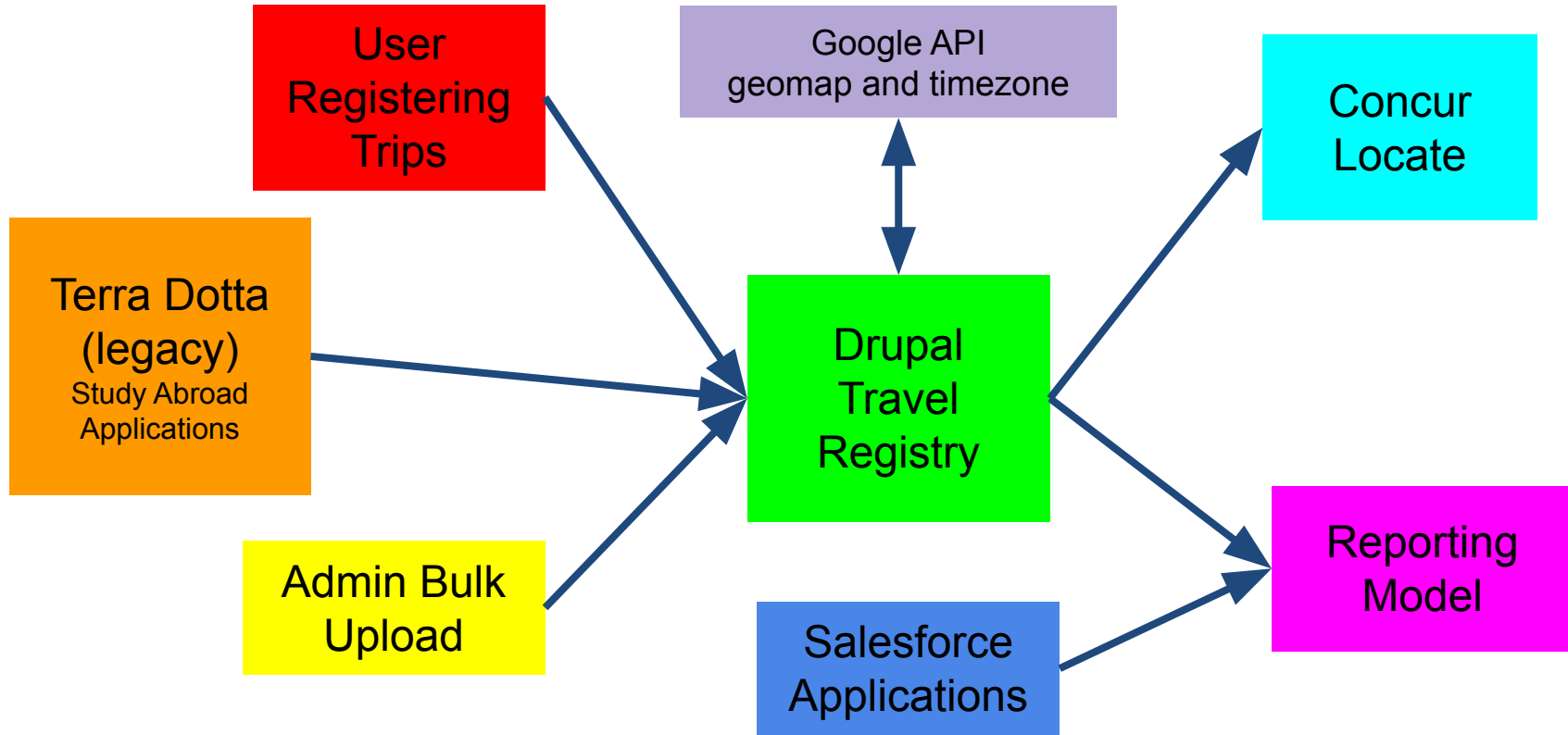- Geolocation
- Conditional Fields
- Views

Live Demo

https://test-travel-reg.pantheonsite.io[...]uate-trip

Travel Registry - Architecture

Travel Registry

Multiple points of integrations
1. Legacy data upload via CSV and business logic to convert data. (Terra Dotta)
2. Custom bulk upload form for mass uploading trips into Drupal by site Admins.
3. Google API - geolocation and timezone.
4. API (JSON) connection to Concur Locate for emergency mass notification.
5. JSON API feed to Reporting Model.

CSV upload

- Terra Dotta is being retired and replaced by Salesforce.
- There is no endpoint in Terra Dotta to continuously feed the data out.
- Terra Dotta admins are able to run exports of data to CSV
- Built custom module that could manipulate the raw csv data and import this into Drupal.
- While Terra Dotta is being phased out this is how data will flow from Terra Dotta to the Travel Registry

Bulk upload

- There are many large groups that travel at Cornell that are not associated with a study abroad program.
- In order to get this data into the Travel Registry and through to Concur Locate we created a bulk upload form.
- This form takes all the information about the travellers on these group trip and their location and generates records that are then sent to Concur Locate.

Google API

- In order to consistently identify a travelers whereabouts we need to pass the timezone of the location the traveler is travelling to through the Concur API.
- For reporting purpose we need to identify the travelers country and city. The geofield module does not provide this when filling in the location field.
- Our integration runs on the saving of a node so it works with all of our other integrations.
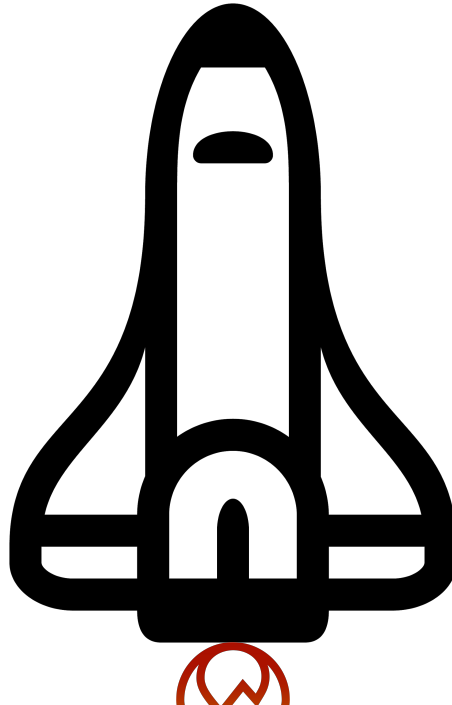
API Connection to Concur Locate

- Concur Locate is Cornell's mass notification system that allows the send of automated emails and texts when incidents occur in locations abroad
- VIA a JSON api we take any data that is entered into the Travel Registry and in real time update Travellers location data in Concur Locate
- For mass uploads (CSV and Bulk) we could not do this in real time due. Passing and parsing that much data needed to be batched and sent over on cron.

Successes with Implementation

- Use of Drupal content types to gather trip information.
- Multiple integrations to allow for data to flow from multiple sources into Concur Locate.
- Successful integration with a reporting model to allow user ad hoc reporting.

Space Management

Purpose

To help facilitate in the allocation and approval of physical spaces on campus to be used for meetings, parties, graduation and all other gatherings that require campus endorsement.

25Live vendor system

- 25Live is an excellent class scheduling and space assignment system.
- 25Live could not meet Cornell's need for our event registration form. It was not possible to add the branching and business logic associated with this process in 25Lives scheduling system.
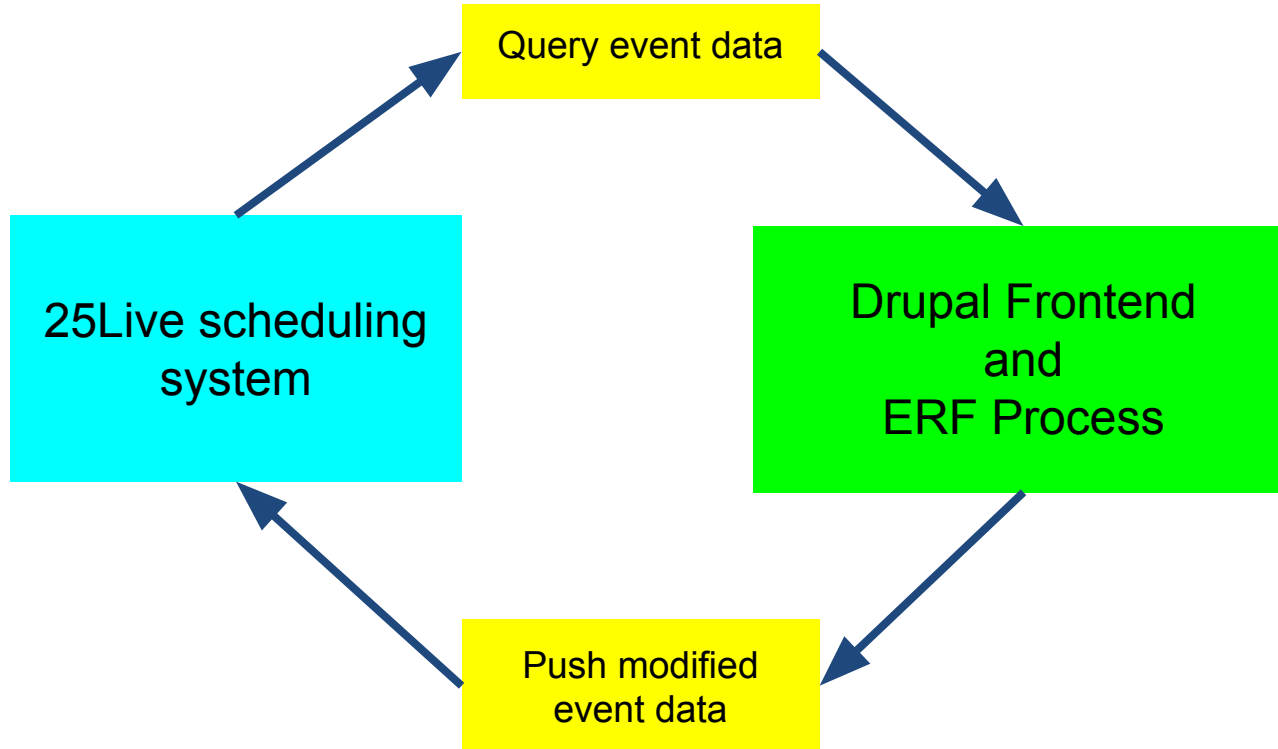
Drupal Architecture

- Webform
- Contextual
- Contextual
- Contextual

Live Demo

2019-AAAAMR

Space Management - Architecture

Workflow and our integration

- When a user comes to the ERF they are prompted to enter their event ID. This queries 25Live through its API and gets the event information as XML.
- We check this information to see if a user has access to submit an ERF or if an ERF has already been submitted for the given event.
- A series of preconditions are checked and flagged.
- We propulate some of the key questions asked in 25Live which will then allow the user to follow branching logic through the ERF webform.

Workflow and our integration

- Upon the saving of a webform submission a series of custom logic is run
- Based on the answer to the webform results a series of notifications or approval tasks are added to the event along with all of the responses in the webform.
- Once we have correctly updated the event XML with the answer we pass this information back to 25Live.

Workflow and our integration

- If we successfully pass the information our process is complete and 25Live takes over.
- If not we requeue the user's answer to push across on cron. There are a few reasons this would fail to push the primary one is if someone is editing the event in 25Live at the time we are trying to push back our XML edits.

Successes with Implementation

- How we configured the webform and the use of conditional fields allow for business logic to be stored in configuration.
  - As long as there are no major changes this can be done through Drupal config.
- The backend code has been simplified to react to the webform submissions results instead of hardcoding in values
  - Use of custom config entities allow for config over hard coding values.

Questions?

Nick Tubbs
- Email: nt328
- Everywhere else: sbbutkcin

Bill Juda
- Email: wfj24@cornell.edu
- Github: judaw