# Technical Report: The Austinites

Sheeyla Garcia Jesus Hernandez Kyle Nicola
Stephen Ridings Carlos Rodriguez Mark Sandan

[splash page]

## Contents

# 1 Introduction

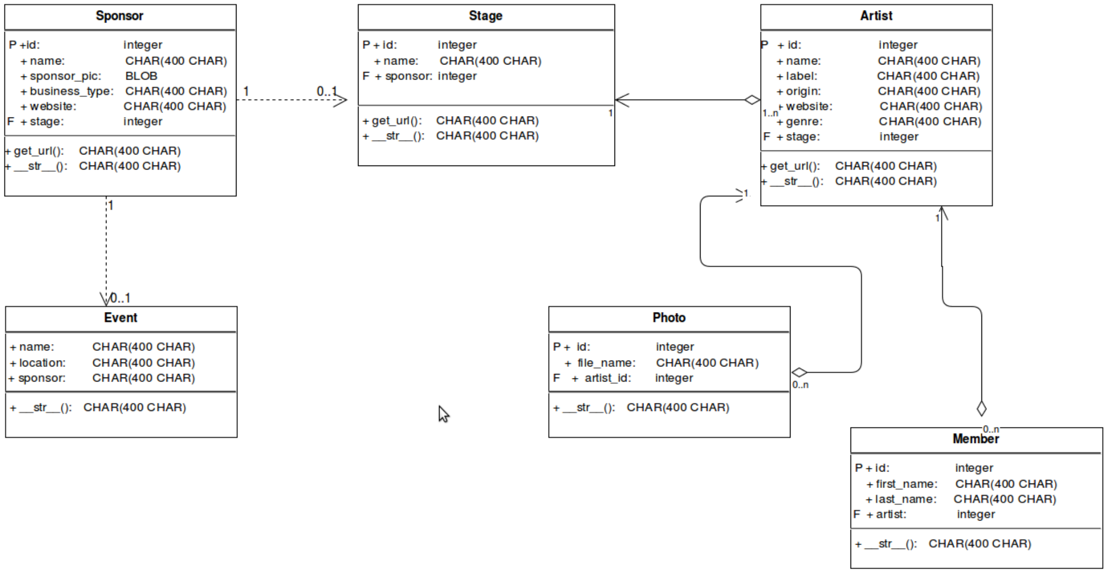
The website we are designing is about the 2014 Austin City Limits (ACL) music festival. The website design has three main pages: Artists, Stages, and Sponsors. The website allows anyone to view pages about current Artists, Sponsors, and Stages. The technologies used are PythonAnywhere, Python 3.4, Django 1.6, Twitter Bootstrap 3.2, Apiary, and the database sqlite3. PythonAnywhere is used to host the site using the Django web-framework to construct the necessary models and views to handle HTTP requests. Twitter Bootstrap is used to organize common data among different types of pages into a base template HTML file.

The database we chose to test with (see Tests) is sqlite3. We currently don't use it to load any data to render the HTML pages.

# 2 Design

Our current design of the website uses HTML and Twitter Bootstrap to stylize each page and PythonAnywhere to host the page. Using Django's templating language we are able to reuse html files by extending from them. Currently we only have a single base html page that uses Twitter Bootstrap. We have nine pages in static HTML that provide examples of the web page design and content.

[UML model here]


Sponsor
P +id: integer
+ name: CHAR(400 CHAR)
+ sponsor_pic: BLOB
+ business_type: CHAR(400 CHAR)
+ website: CHAR(400 CHAR)
F + stage: integer
+ get_url(): CHAR(400 CHAR)
+ __str__(): CHAR(400 CHAR)
1
0..1
Stage
P + id: integer
+ name: CHAR(400 CHAR)
F + sponsor: integer
+ get_url(): CHAR(400 CHAR)
+ __str__(): CHAR(400 CHAR)
1
Artist
P + id: integer
+ name: CHAR(400 CHAR)
+ label: CHAR(400 CHAR)
+ origin: CHAR(400 CHAR)
+ website: CHAR(400 CHAR)
+ genre: CHAR(400 CHAR)
F + stage: integer
+ get_url(): CHAR(400 CHAR)
+ __str__(): CHAR(400 CHAR)
1..n
1
1
1
0..1
Event
+ name: CHAR(400 CHAR)
+ location: CHAR(400 CHAR)
+ sponsor: CHAR(400 CHAR)
+ __str__(): CHAR(400 CHAR)
Photo
P + id: integer
+ file_name: CHAR(400 CHAR)
F + artist_id: integer
+ __str__(): CHAR(400 CHAR)
0..n
0..n
Member
P + id: integer
+ first_name: CHAR(400 CHAR)
+ last_name: CHAR(400 CHAR)
F + artist: integer
+ __str__(): CHAR(400 CHAR)

## 2.1 App Design

The current directory structure on PythonAnywhere is outlined below for reference to files:

[tree -d layout here]

## 2.2 Web Pages

Each web page has basic information about a particular artist, sponsor, or stage involved in the ACL music festival. Each page includes a bio, video, official website, Youtube channel, a Twitter feed, and Picture.

### 2.2.1 Artists

Artist Pages can be reached from the home page or from Stage pages depending on whether the Artist played on a Stage that was hosted by a Sponsor.

### 2.2.2 Sponsors

Sponsor pages can be reached from the home page or from

### 2.2.3 Stages

[Example Relation scheme here]

## 2.3 RESTful API

The REST API for the website currently documents only GET

## 2.4 Django Models

The Django models created represent the entities we intend to keep in the database for the future when use dynamically loaded pages. The following subsections document the attributes and intended functionality of each class instance method.

### 2.4.1 Artist

### 2.4.2 Sponsor

### 2.4.3 Stage

### 2.4.4 Photo

The Photo class extends from the Django models.Models class and has two main attributes:

```
file_name
artist
```

The Foreign key is the artist that the Photo is associated with. Thus the Artist is a parent of a one-to-many relationship with a Photo. Currently the only method defined is

`__str__`

### 2.4.5 Event

### 2.4.6 Member

Detailed documentation of each of our Django models

# 3 Tests