Technical Report: The Austinites

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

July 24, 2014

Contents

Figure 1: Splash Page for the second iteration of IDB2 showing the group name and the buttons in the three main pages.

1 Updates

Updates for the second iteration at a glance:

- the splash page and web design has been $fanciedup$
- Dynamic webpages supporting our models instead of static webpages
- MySQL is being used currently as the database backend
- Django Rest framework installed to implement the API
- Refactored the Models used to include a Media models. Photo and Member models have been deprecated.
- Django commands have been installed to load data from formatted text files for Artist, Stage, and Sponsor

1.1 Summary

The main changes that have been made can be summarized into three categories: dynamic functionality, REST implementation, and the model refactoring.

To implement a dynamic webpage we are currently using html,css,javascript, and the django template language to access instances of Artists, Sponsors, and Stages to fill the

content of the webpages. The REST implementation required installing the django rest framework on the pythonanywhere envioronment which enabled us to return JSON representations of MySQL database instances. The tests for the REST implementation have also been added to the test suite bringing the total number of tests to ???.

The models have been refactored to include media classes that provide the dynamic content for the Artist, Sponsor, and Stage models when they are loaded into the webpage. The media content contains embedded links to youtube, facebook, twitter, etc.

Database loading has been relegated to using python scripts which have been installed as django commands used with manage.py. They are located in the $mysite/cs373/management/comman$ directory.

2 Introduction

Our website is about the 2014 Austin City Limits (ACL) music festival, an annual three-day American music festival that takes place on 46-acres in Zilker Park located in Austin, Texas. The website design has three main pages: Artists, Stages, and Sponsors along with a splash page. An Artist is allowed to perform on one stage but stages can have many artists perform on them. A sponsor is a business entity that may or may not sponsor a stage. A stage is a physical location in the ACL festival that many artists play on. It can have only one sponsor that sponsors it for any given year. Artists and Sponsors are related through Stages.

The website allows anyone to view pages about the current Artists, Sponsors, and Stages involved in the festival. A user can find links from a specific artist page to the stage they're playing on as well as the sponsor sponsoring the stage. Similarly for the stage and sponsor pages. The structure is modeled after the IMDB website (http://www.imdb.com/) where the entities are highly coupled. A problem that we are facing is that the information we need to complete the project hasn't been published as of the date this report except the Artist lineup. Their scheduled performances which include information should be set to release sometime this month.

2.1 Technology Stack

The technologies used are:

- PythonAnywhere: a web hosting service with python environments supported. Currently we use Python 3.4 and Django 1.6+.
- Twitter Bootstrap 3.2: a web hosting service with python environments supported Python 3.4, Django 1.6.
- Apiary: an online service to provide an API for client-side web access to our databases.
- MySQL: The current database backend using the mysql.connector.django engine.

3 Design

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 is extended from and uses

Twitter Bootstrap. We have 24 pages that are dynamically loaded. The design and content of each page is outlined below.

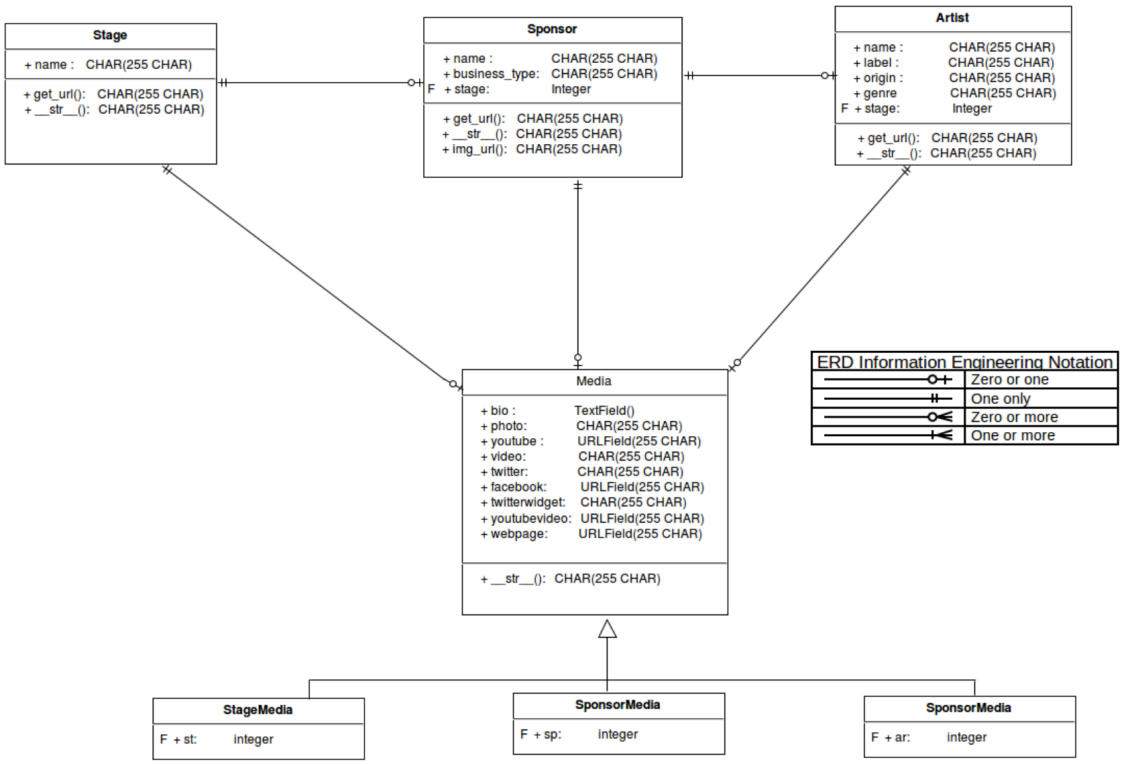

Figure 2: The current UML schema depicting the relationships between the Django models. The main three models are Artist, Sponsor, and Stage each of which have an associated Media page.

3.1 Web Pages

Each web page has basic information about a particular artist, sponsor, or stage involved in the ACL music festival. All pages will include a navigation bar at the top of their page that will allow the user to go back to the main "splash" page, as well as reach the Artists, Sponsors, and Stages pages. In future phases we are considering incorporating a search bar inside of the navigation bar, so that the user can search all categories. Mobile browsers are able to view the webpage correctly and set the height and width to a percentage of the size of the screen.

3.1.1 Splash Page

- URL: https://theaustinites.pythonanywhere.com/

The "splash" page will be the first page a visitor to the site will see. It will provide button style links to all subcategories (Artists, Sponsors, Stages). Currently the page lists the group name and group members.

3.1.2 Artists

- URL: https://theaustinites.pythonanywhere.com/artists

Artist Pages can be reached from the home page as well as from the Stage or Sponsor pages depending on whether the Artist played on a Stage that was hosted by a Sponsor.The page currently lists the dynamic web pages for music artists:

- Outkast

Each Artist page includes the Artist name, an Artist photo, the label, the origin, genre, the sponsor,the stage, a bio, the official website to the sponsor, a facebook page, a bio, youtube video, youtube channel, and twitter feed.

3.1.3 Sponsors

- URL: https://theaustinites.pythonanywhere.com/sponsors

Sponsor pages can be reached from the home page as well as from the Artist or Stages pages depending on whether the Sponsor hosted a Stage, and that particular Artist played on that Stage. The page currently lists ACL festival sponsors:

- Honda
-

Each Sponsor page includes the Sponsor name, a sponsor logo, the origin, stage sponsored, the artists playing the stage, the official website to the sponsor, a facebook page, a bio, youtube video, and twitter feed.

3.1.4 Stages

- Location: https://theaustinites.pythonanywhere.com/stages/

Stage pages can be reached from the Home page (splash) as well as from the Artist or Sponsor pages depending on whether the Sponsor hosted a Stage, and that particular Artist played on that Stage. The page currently lists the following stages:

-
-

Each Stage page includes the Stage name, a Stage logo, the artists playing the stage, the official website to the sponsor, a facebook page, youtube video, and twitter feed.

3.2 RESTful API

The API allows GET requests to the following models: Stages, Sponsors, Artists. The Members and Photos models have been deprecated. The following section will detail the attributes for the modules, and how the server will respond to the GET requests. See the Links section below to view the Apiary API.

3.2.1 Stages

When a GET is called on $[/stages]$ it will return a JSON representation of all the Stages in the database. It will be a list of the stages along with their attributes. When a GET is called on $[/stages/id]$ it will return a JSON representation of a single Stage in the database with the given id. It will list the stage and all of it's attributes
Example of single stage:

```
{
        "id": 42,
        "name": "Stage name"
}
```

3.2.2 Sponsor

When a GET is called on $[/sponsors]$ it will return a JSON representation of all the Sponsors in the database. It will be a list of the sponsors along with their attributes. When a GET is called on $[/sponsors/id]$ it will return a JSON representation of a single Sponsor in the database with the given id. It will list the sponsor and all of it's attributes
Example of single sponsor response:

```
{
        "id": 42,
        "name": "Sponsor name",
        "business_type": "Type of Business",
        "website": "URL of sponsor website",
        "stage": 12
}
```

3.2.3 Artist

When a GET is called on $[/artists]$ it will return a JSON representation of all the Artists in the database. It will be a list of the artists along with their attributes. When a GET is called on $[/artists/id]$ it will return a JSON representation of a single Artist in the database with the given id. It will list the artist and all of it's attributes.
Example of single artist:

```
{
        "id": 42,
        "name": "Artist name",
        "label": "Label of artist",
        "origin": "Where the artist was from",
        "website": "URL to the artist",
        "genre": "Genre of the artist",
        "stage": 22
}
```

3.3 Django Models

The Django models created represent the entities we believe are essential to modeling the ACL festival. The following subsections document the attributes and intended functionality of each class instance method. The following sections use $child$ and $parent$ in the sense of the schema relationship depicted by the UML diagram in Figure 2 and not in the sense of the object oriented inheritance (the only object each model extends is models.Model except the Media classes ArtistMedia, SponsorMedia, and StageMedia that inherit from Media). Currently there are a total of six class.

3.3.1 Artist

The Artist class represents the current Artists playing on a sponsored stage. All Artists will be a child of some stage depending on whether they are playing that Stage or not. The entity relation between Sponosor and Artist The Artist class is implemented using the following:

```
attributes:
- name: the name of max length 255 characters.
- label: the artist label with a maximum length of 255 characters.
- origin: the place of origin the artist/group formed with a maximum length of 255 cha
- genre: the genre associated with the artist. May span more than one. maximum length

- stage: Foreign Key, integer of type field.

methods:
- get_url(): returns the string "/artists/{id}" . Maximum number of 255 characters.
- __str__(): returns the name string of the artist.Maximum  number of 255 characters
```

3.3.2 Sponsor

The Sponsor class represents the ACL sponsors that sponsor a Stage for the Artist to perform on. This relationship is expressed using the one-to-many relationship between the Sponsor and the Stage class.

The Sponsor class has the following attributes and methods:

```
attributes:
- id: Primary Key, integer type field
- name: a character type field with a maximum length of 255 characters
- business_type: a character type field with a maximum length of 255 characters
- stage: Foreign Key, integer of type field

methods:
- get.url(): returns the string "/sponsors/{id}". Maximum number of 255 characters.
- __str__(): returns a string that represents the name of the sponsor.
```

3.3.3 Stage

The Stage class is meant to represent the stage that an Artist will perform on. All stages have one sponsor. The Stage class extends from the Django models.Models class.

The Stage class has the following attributes and methods:

```
attributes:
- id: Primary Key, integer type field
- name: a character type field with a maximum length of 255 characters


methods:
- get.url(): returns the string "/stages/%s/{name}" where name is the stage name.
- __str__(): returns a string that represents the name of the stage
```

3.3.4 Media

The Media class represents all media informaton that is associated with a sponsor, artist, or stage.

```
attributes:
-id: Integer
-bio: A CharField of 255 characters. Biography.
-Photo: A CharField of 255 characters.
-Youtube: A CharField of 255 characters. url for Youtube Channel
-Video: A CharField of 255 characters.
-Twitter: A CharField of 255 characters. url for Twitter
-Facebook:A CharField of 255 characters. url for Facebook
-twitterwidget: A CharField of 255 characters.
-webpage:

methods:
- __str__(): returns the file_name of the photo.
```

4 Unit Tests

The unit tests currently implemented reflect the expected functionality of the django models and database. Currently we are using the sqlite3 database as a backend. We assume for each function in the set of tests that the state of the database is reset. There are currently 30 tests with at least 4 tests for each model. The following give descriptions of each model test.

4.1 Stage Test

Total tests: 7

- test_create_empty_stage: Tests that uninitialized (stateless) Stages can be created.
- test_create_stage : Tests that the $name$ attribute is set correctly when initialized.
- test_get_stage: Tests that the get_stage method returns the Stage instance we saved into the database. Also tests the filtering functionality of setting the id of the instance.

- test_stage_name: Tests the max_length property of CharField when setting the name of Stage.
- test_stage_url_1, test_stage_url_2,test_stage_url_3: Tests whether an instance of the desired Stage class is saved in the database table and returns the correct url attribute.

4.2 Artist Test

Total tests: 8

- test_create_artist: Tests the Artist instance attributes actually match the given values that were passed when constructed (including a Stage instance)
- test_get_artist: Tests that an instance of Artist can be retrieved from the database with the attributes used to create it.
- test_empty_artist: Tests that an IntegrityError is thrown when an Artist instance is saved in the database without setting a ForeignKey.
- test_empty_artist_with_stage: Tests the default values of an Artist instance with a ForeignKey.
- test_artist_name: Tests the max_length property of CharField when setting the name of Artist.
- test_artist_url_1, test_artist_url_2, test_artist_url_3: Tests that the many-to-one relationship between Artists and Stages is valid. Also tests the get_url method returns the correct url according the id of the instance.

4.3 Sponsor Test

Total tests: 6

- test_empty_sponsor: Tests that a Sponsor's default values and ID are set correctly.
- test_save_sponsor: Tests that the name matches the passed value given when the Sponsor instance was created.
- test_get_sponsor_name: Tests the max_length property of CharField when setting the name of Sponsor.
- test_sponsor_url_1,
- test_sponsor_url_2: Tests that the get_url is returned in the proper format.

4.4 API Test

The REST API is tested using the django rest framework. The django rest framework has a built in library that allows the tests to mimic API clients that interact with the API. Currently only GET requests are allowed. Any other request will throw ??? .

- test_empty_sponsor: Tests that a Sponsor's default values and ID are set correctly.
- test_save_sponsor: Tests that the name matches the passed value given when the Sponsor instance was created.
- test_get_sponsor_name: Tests the max_length property of CharField when setting the name of Sponsor.
- test_sponsor_url_1,
- test_sponsor_url_2: Tests that the get_url is returned in the proper format.

5 Expansions

This section outlines probable features and functions for the next release.

- More links to more social media accounts (Instagram or SoundCloud) for Artist pages.
- Stage pages will link to a map of the festival so users can see their location.
- Sponsor models might also include include an attribute to store business content such as the location of their headquarters, the name of their respective CEOs, stocks, etc.
- Tests that will check that the database is correctly populated with its new attributes as well as test all the new methods included.
- An updated API that will include the option to make put requests so that data can be uploaded using django forms.

5.1 Third Iteration extension

For the third iteration we are planning to extend the concept of the website to cover years previous to 2014. This would allow the user to have more expressive queries. For example a user might want to know how many times an artist has played on a stage or was sponsored by some business, the distribution of businesses by industry that have sponsored certain genres, or simply a list of all hip hop artists that have played in ACL from a certain year. The propsed change would also enhance the model relationships to become many to many. For example an artist may play on the same stage from previous years that were sponsored by different businesses. Also an artist can now play on more than one stage over time. The inclusion of past ACL artists, sponsors, and stages will also allow for more data of each model as a history accumulates. This is our strongest reason for the propsed change because the schedule for 2014 has not been published yet and it

may take some time before it does get published. Implementing this proposal will address our curent problem. We plan on including timelines for each Artist, Sponsor, and Stage page to take advantage of this extension.

6 Links

- Home Page: https://theaustinites.pythonanywhere.com/
- Apiary API: http://docs.aclprojectapi.apiary.io/
- Github: https://github.com/sandan/cs373-idb
- http://www.aclfestival.com/