Band Tracker Design Document

Udacity iOS NanoDegree – Project 5

Author	Johan Smet
Revision	v0.1
Last modified	2015-10-28
Number of pages	24

Table of Contents

1. Revision History

Revision	Date	Comments
v0.1	2015-09-02	Initial draft

2. Introduction

After a few years of frequenting music festival it gets difficult to remember which bands you've seen and whether you liked them or not. This apps allows you to keep track of the performances you've seen and what you thought of the show. Additionally it will assist you in choosing which bands to see when going to an event.

3. Server architecture

3.1. Overview

The server is a Node.js webservice that handles client-requests through a rest-like API. The service written in TypeScript using Express and has a MongoDB database as the storage back-end.

The server provides basic information to the application. At the moment only an administrator can add or update data, the database is read-only to the clients. In the future this will change when more social network related function are implemented.

3.2. Database design

A NoSQL document database, specifically MongoDB, was chosen to implement the server back-end. This has influenced the design of the data structures. Please refer to Illustration 1: Database Schema (page 3) for a schematic overview.

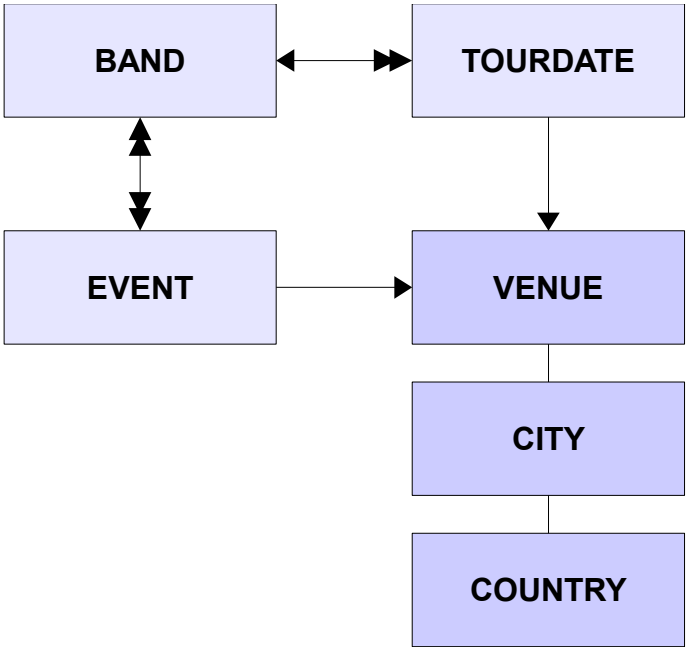


Illustration 1: Database Schema

3.2.1 Datatypes

- number : signed integer or decimal number (decimal separator = point)
- string : UTF-8 encoded string
- localized string : string in multiple languages (dictionary of language-code + value)
- date : using ISO 8601 format (yyyy-mm-dd'T'hh:mm:ss.sss)
- boolean : true / false

3.2.2 Band

```
{
      MBID : <string>,                        // MusicBrainz ID
      name : <string>,                        // Name of the band
      genre : <string>,                       // genre of music
      imageUrl : <string>,                    // url of an image of the band
      biography : <localized-string>,         // Short biography of the band
      source : string                         // Reference to data source
}
```

3.2.3 TourDate

```
{
      bandId : <string>,                      // MusicBrainz ID
      startDate : <date>,
      endDate : <date>,
      stage : <string>,                       // optional
      venue : <string>,                       // optional
      city : <string>,                        // optional
      countryCode : <string>,
      supportAct : <boolean>                  // was this gig as a support act
}
```

3.2.4 Event

```
{
      description : <string>,
      startDate : <date>,
      endDate : <date>,
      venue : <string>,                       // optional
      city : <string>,                        // optional
      countryCode : <string>,
      performances :
      {
          <date> :                            // day
          {
                  <string> :                        // stage
                  [
                      {                             // gig
                        bandId : <string>,
                        startTime : <date>,
                        endTime : <date>
                      },
                      ...
                  ],
                  ...
          },
          <date> :
          {
                  ...
          }
                ...
      }
}
```

3.2.5 Venue

```
{
	name : <string>,
	city : <string>,
	countryCode : <string>
}
```

3.2.6 City

```
{
	city : <string>,
	countryCode : <string>
}
```

3.2.7 Country

```
{
	code : <string>,
	name : <localized string>
}
```

3.3. API reference

3.3.1 Bands

/bands		GET
Action	Retrieves a list of all known bands	
Permission	Blessed application	
Parameters	-	-
Response-body	*list*	List of band records

/bands/{id}		GET
Action	Retrieve information about a particular band	
Permission	Blessed application	
Parameters	id	The MBID of the band to retrieve.
Response-body	*record*	The current contents of the BAND-record with the specified ID.

/bands		POST
Action	Add a new band or update an existing band. The server retrieves extra information about the band from external services and returns it to the client.	
Permission	ADMIN	
Request	*record*	The data of the band to add or update
Response-body	*record*	The current contents of the BAND-record with the specified ID.

/bands/find-by-name?name={pattern}		GET
Action	Search for bands matching a particular name.	
Permission	Blessed application	
Parameters	pattern	The name or pattern to search by
Response-body	*list*	List of BAND-records that match the query. Only a subset of available fields are returned.

:w

3.3.2 TourDate

/tourdate			POST
Action	Add a new tourdate or update an existing record.		
Permission	ADMIN		
Request-body	*record*	The tourdate to be added/updated	
Response-body	*record*	Current information of the TourDate	

/tourdate/find?band={MBID}&start={date}&end={date}&location={location}			GET
Action	Search for tourdates		
Permission	Blessed application		
Parameters	band	The band playing the gig	
	start	(optional) Starting date of the interval to look for tourdates	
	end	(optional) Ending date of the interval to look for tourdates	
	location	(optional) Location of the gig (venue, city or country)	
Response-body	*list*	List of TOURDATE-records that match the query.	

3.3.3 Event

/event			POST
Action	Add a new event or update an existing record.		
Permission	ADMIN		
Request-body	*record*	The event to be added/updated	
Response-body	*record*	Current information of the Event	

/event/{id}			GET
Action	Retrieve details of a specific event		
Permission	Blessed application		
Parameters	Id	Id of the EVENT-record to get more details about	
Response-body	*record*	Current information of the Event	

/event/find?event={pattern}&start={date}&end={date}&location={location}			GET
Action	Search for events		
Permission	Blessed application		
Parameters	name	Pattern to match the name of the event	
	start	(optional) Starting date of the interval to look for events	
	end	(optional) Ending date of the interval to look for events	
	location	(optional) Location of the event (venue, city or country)	
Response-body	*list*	List of partial EVENT-records that match the query.	

3.3.4 Venue

/venue			POST
Action	Add a new venue or update an existing record.		
Permission	ADMIN		
Request-body	*record*	The venue to be added/updated	
Response-body	*record*	Current information of the Venue	

/venue/find?pattern={pattern}&city={city}&country={country}			GET
Action	Search for venues		
Permission	Blessed application		
Parameters	pattern	Pattern to match the name of the venue	
	city	(optional) The city of the venue	
	country	(optional) The country of the venue	
Response-body	*list*	List of VENUE-records that match the query	

3.3.5 City

/city			POST
Action	Add a new city or update an existing record.		
Permission	ADMIN		
Request-body	*record*	The city to be added/updated	
Response-body	*record*	Current information of the city	

`/city/find?pattern={pattern}&country={country}`			GET
Action	Search for city		
Permission	Blessed application		
Parameters	pattern	Pattern to match the name of the city	
	country	(optional) The country of the city	
Response-body	*list*	List of CITY-records that match the query	

3.3.6 Country

`/country`			POST
Action	Add a new country or update an existing record.		
Permission	ADMIN		
Request-body	*record*	The country to be added/updated	
Response-body	*record*	Current information of the country	

`/country/sync?current={hash}`			POST
Action	Retrieve a list of the countries if the specified hash does not match the current hash of the countries on the server		
Permission	Blessed application		
Parameter	hash	Hash of the last synced country list	
Response-body	updated	Boolean to indicate if the list has changed	
	hash	(if updated) hash of the new list	
	list	(if updated) list of COUNTRY-records	

4. iOS App

4.1. Screen flow

4.1.1 The band overview

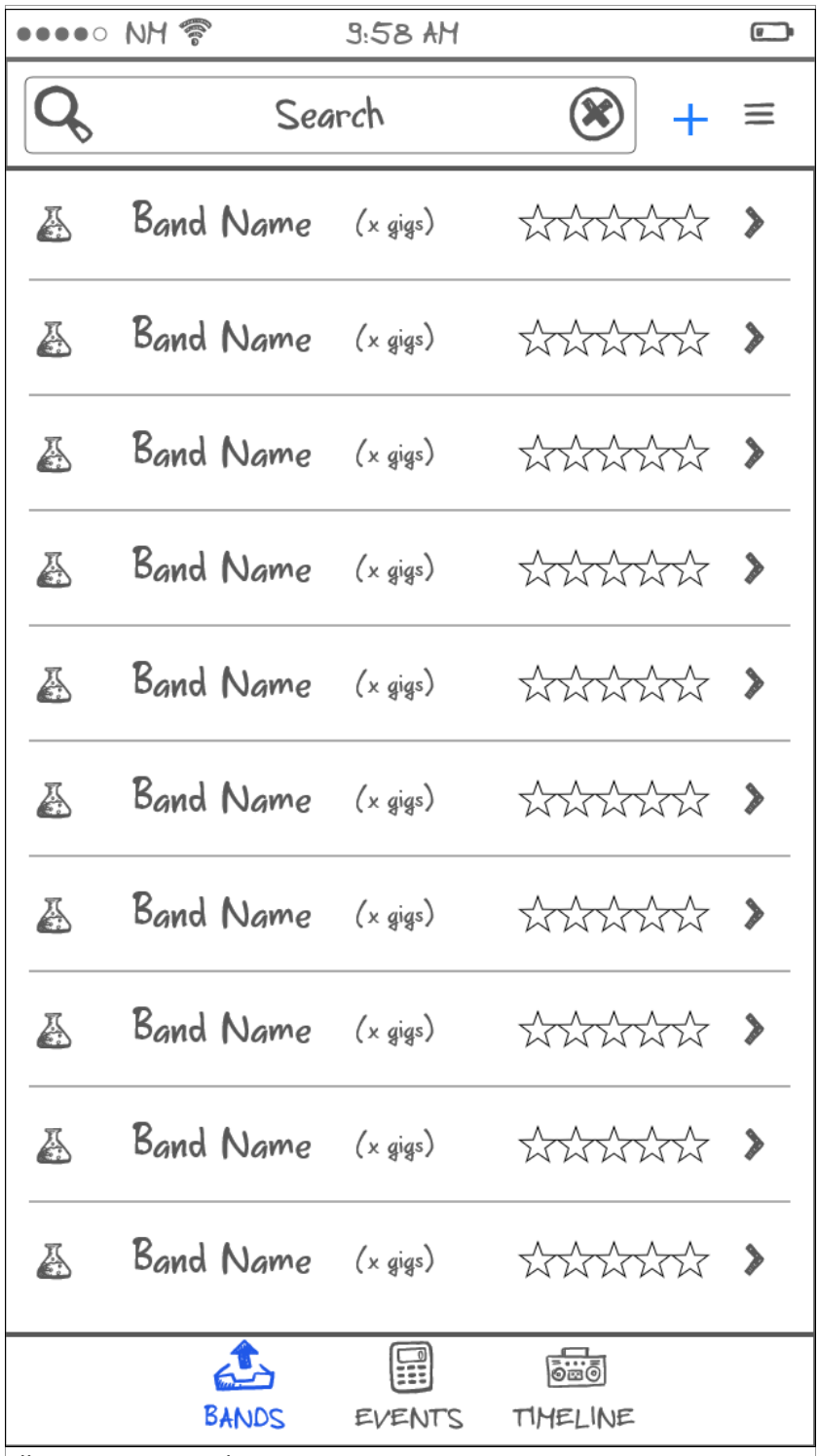


Illustration 2: Band Overview

The band-screen shows an overview of the bands you've seen, sorted by average rating. Tap on a row to see more details. Tap the plus at the top to add a new band.

4.1.2 Band details

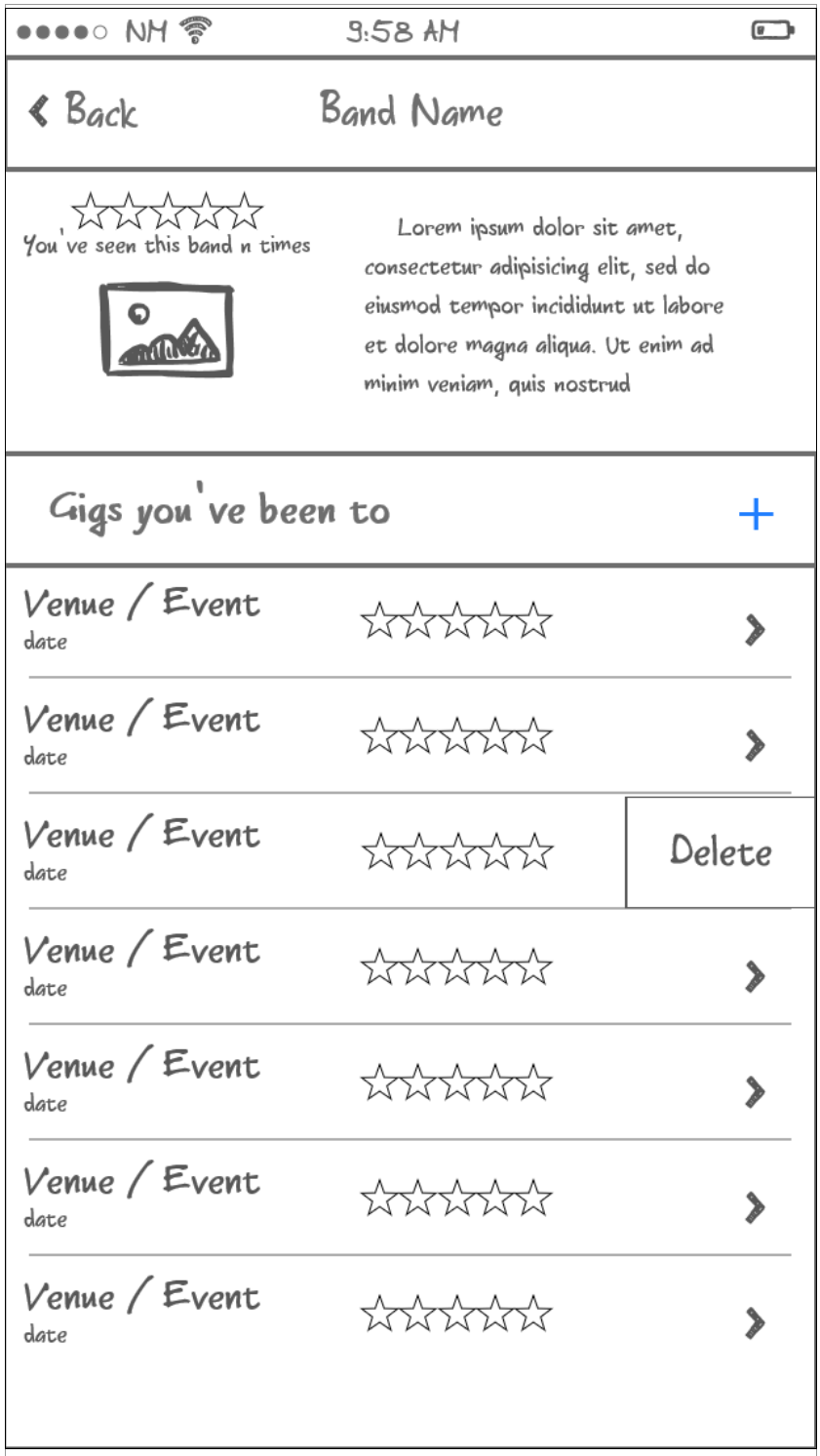


Illustration 3: Band details

The band-details shows some background information of the band and the list of gig's you've them play. Tap a row to see more details of the gig or swipe to delete. Tap the plus to add a new gig.

4.1.3 Guided creation of a gig

When adding a gig to an artist you're first presented with a screen that allows you to search for known gigs on the server. Specify a few parameters and select the wanted gig from the list. If the gig isn't found you can still add it by typing in the details manually.

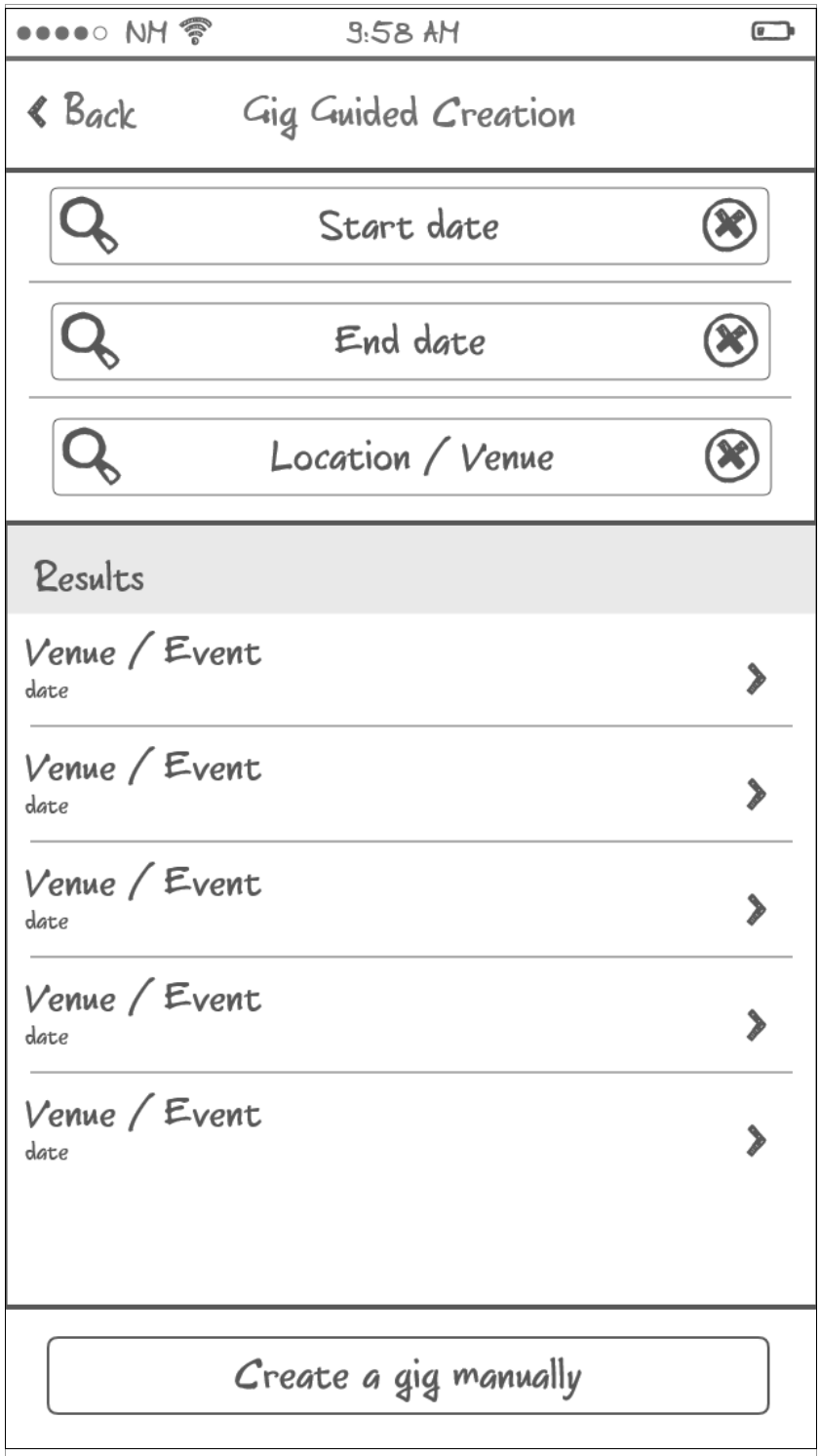


Illustration 4: Guided creating of a gig

4.1.4 Details of a gig

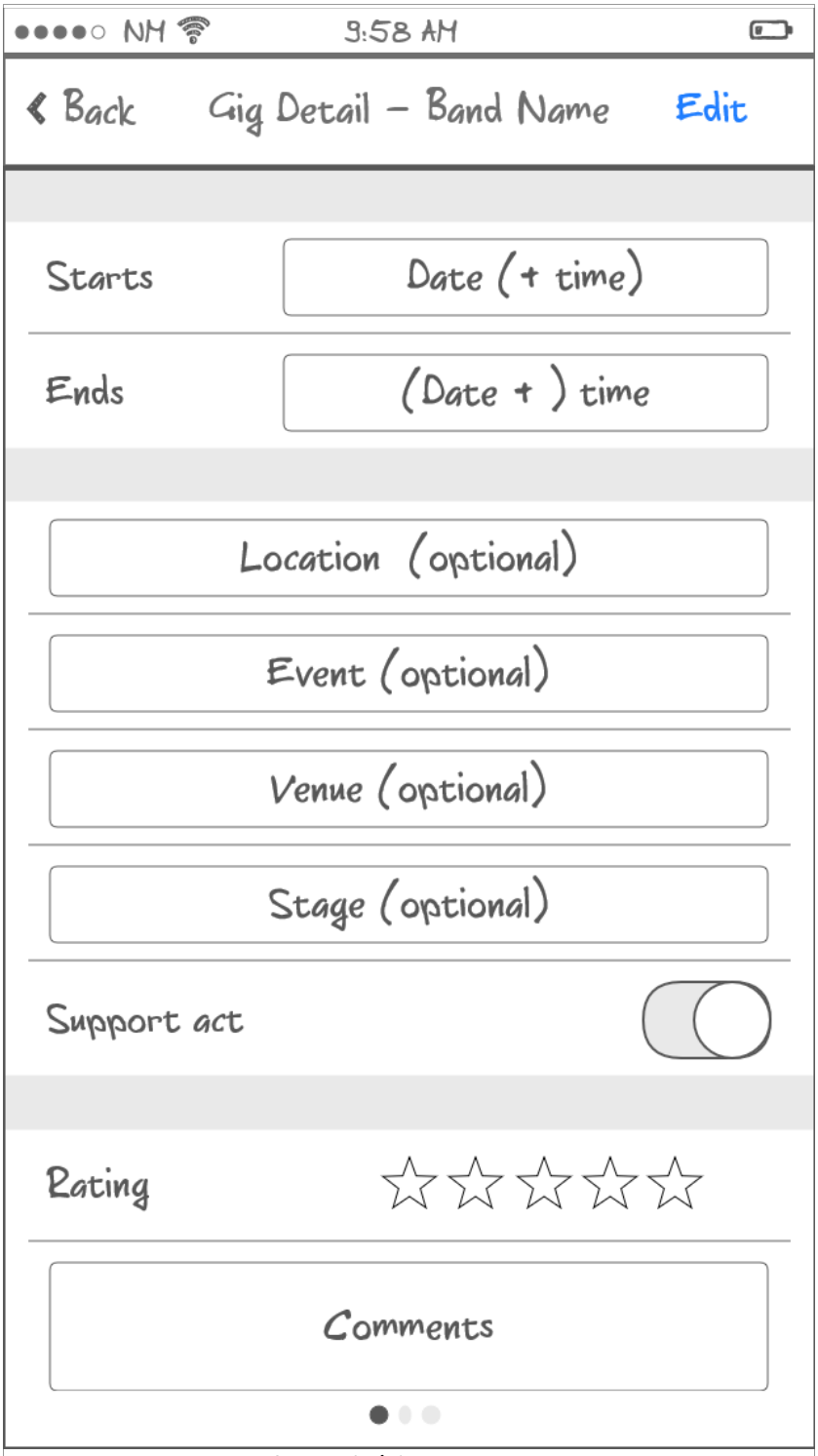


Illustration 5: Details of a gig (1/3)

This screen is used to display details of gig but also to create a new gig. The fields should be self-explanatory. Tap the edit button to change an existing gig. To two following screen are only available when view not creating/editing.

Illustration 6: Details of a gig (2/3)

This screen shows the set list of the gig by fetching the relevant data from setlist.fm.
The third screen lists the top-rated videos of the gig that it finds on YouTube. It either searches for the entire gig or a specific song when the screen is reached by tapping on a song in the set list.

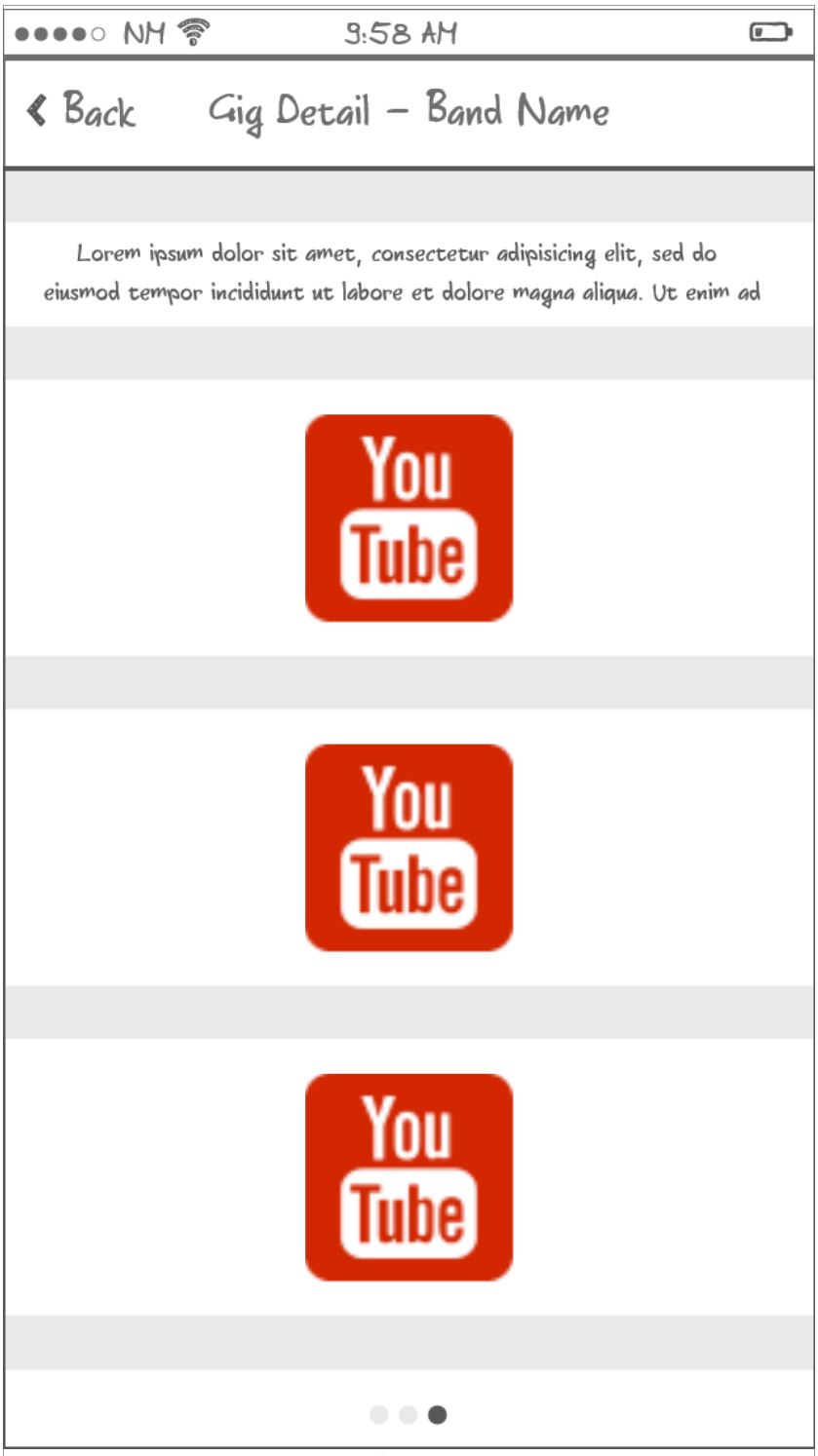


Illustration 7: Details of a gig (3/3)

4.1.5 The event-overview

The event overview shows all the events (festivals or other multi-band events) you've been to. Tap on an event to see more details. Tap on the plus to add a new event.

Illustration 8: Event overview

4.1.6 Creating a new event

The user is first given the opportunity to select a predefined from the server. Enter a few parameters and select the wanted event from the list. When no event matches the one the user is looking for, tapping the “Create an event manually” can be used to create the event by entering all the relevant information.

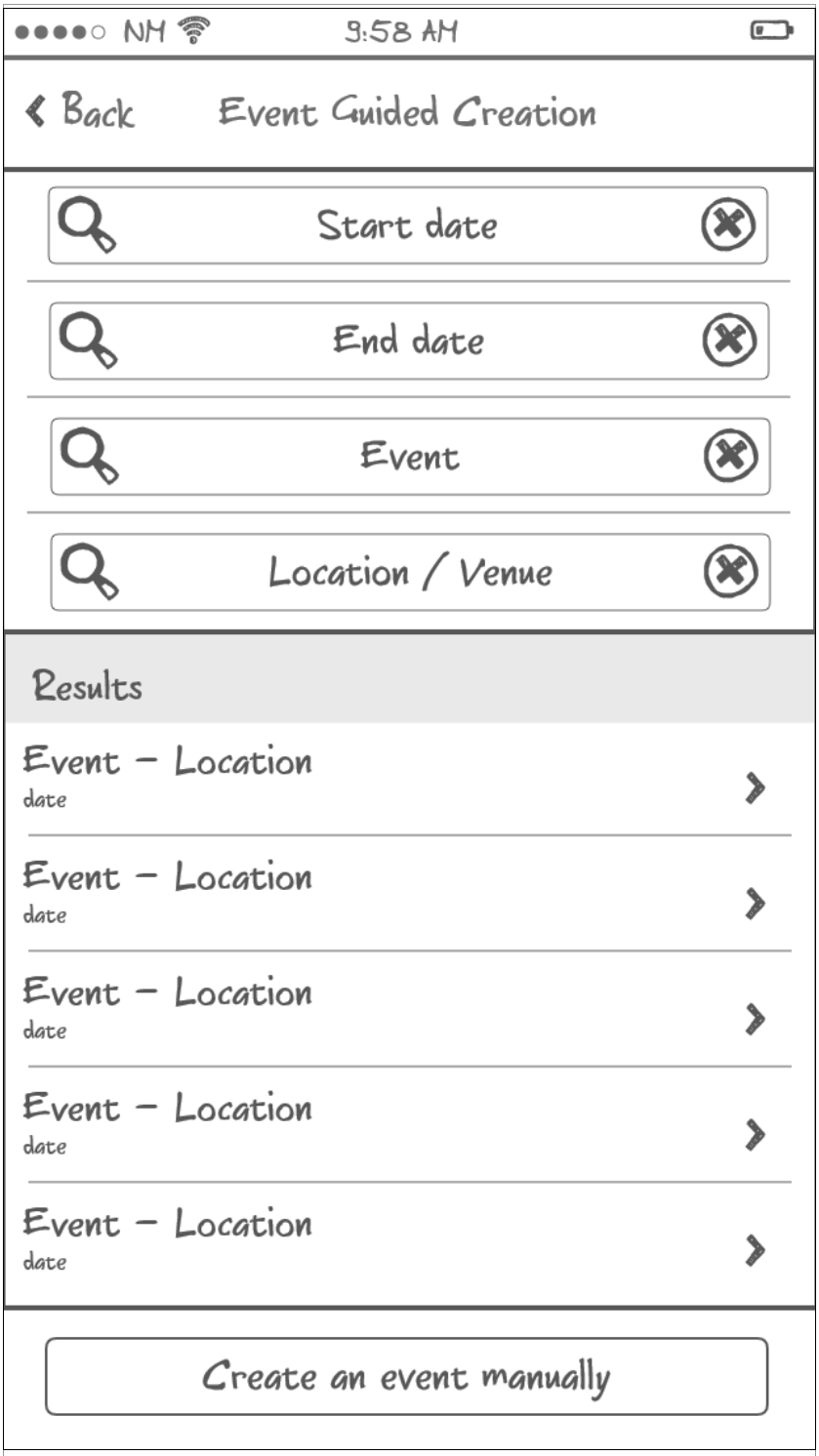


Illustration 9: Guided event creation

4.1.7 Event details

The first screen shows general information about the event. The same screen is used to edit or create an event.

Illustration 10: Details of an event (1/3)

The second screen show the participating bands and allows the user to edit this information.

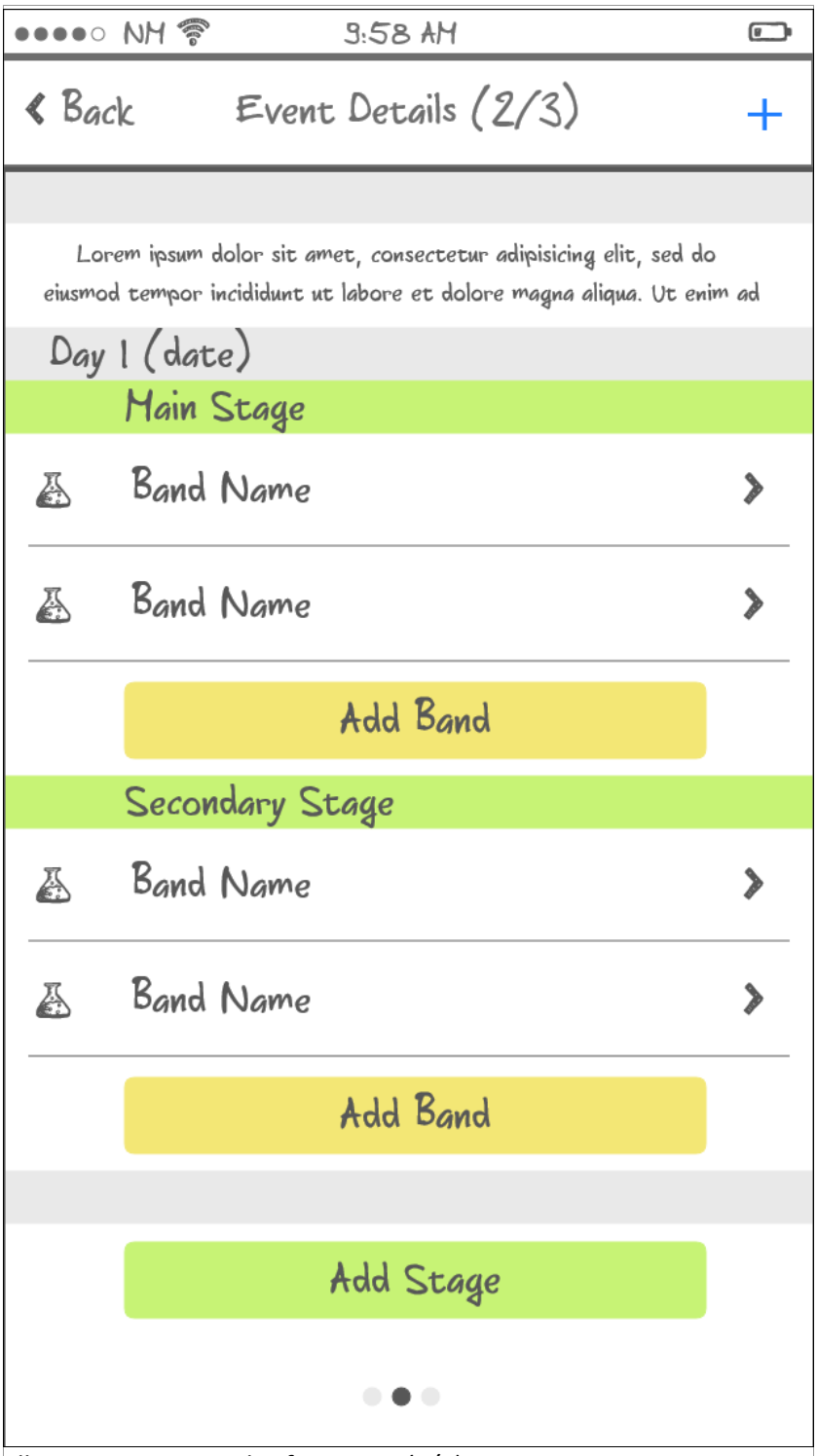


Illustration 11: Details of an event (2/3)

The third screen (not pictured here) shows highest rated YouTube videos of the event.

To add a band to the event the app-standard screen is used. To add a stage a popup windows will be used.

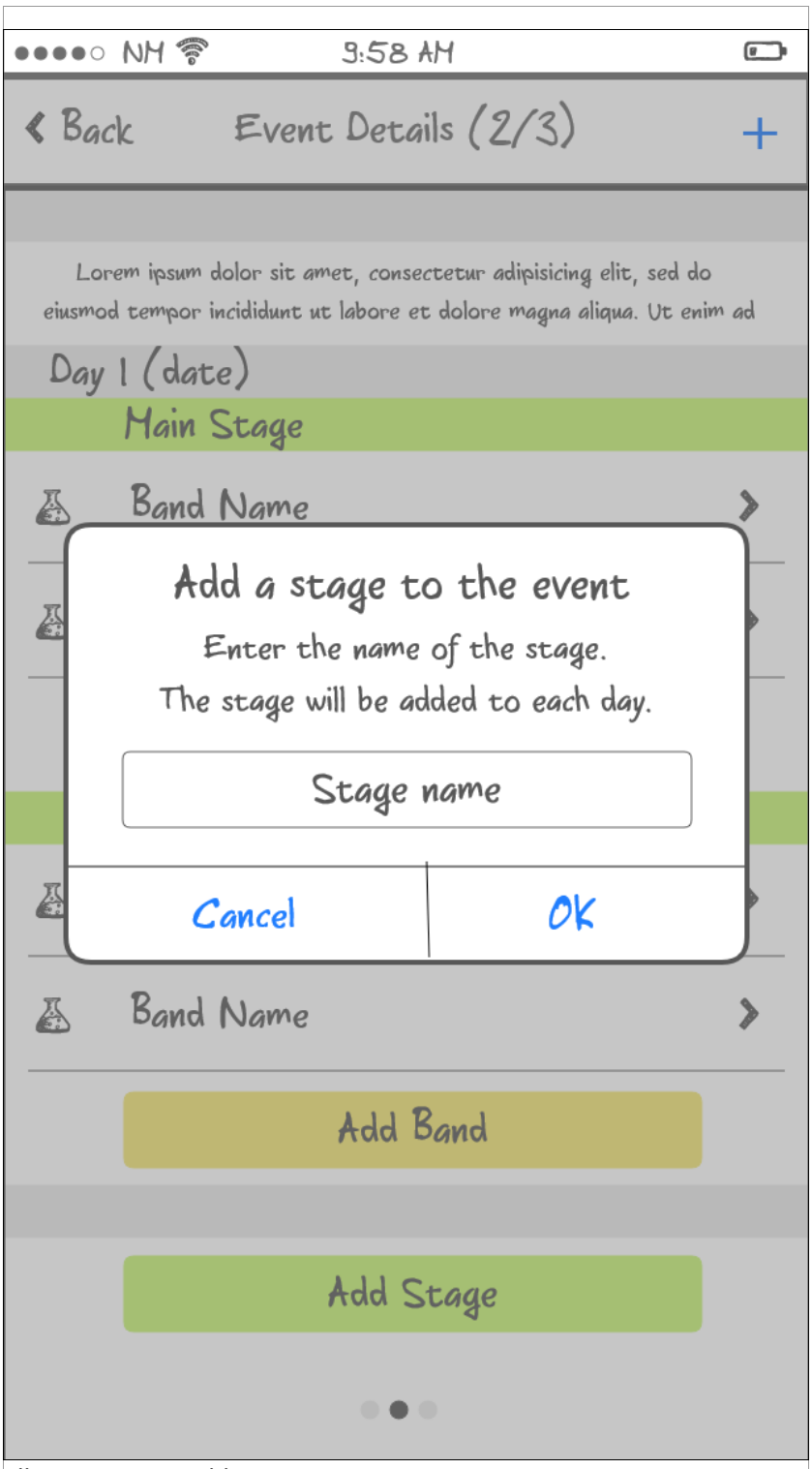


Illustration 12: Adding a stage to an event

The exact flow of adding/editing an event has not been worked out. Sufficient room is left to experiment during implementation to achieve a comfortable work-flow.

4.1.8 Timeline

The last major screen of the app shows a time-line of events and artists you've seen. The list is split into section by month and can be searched.

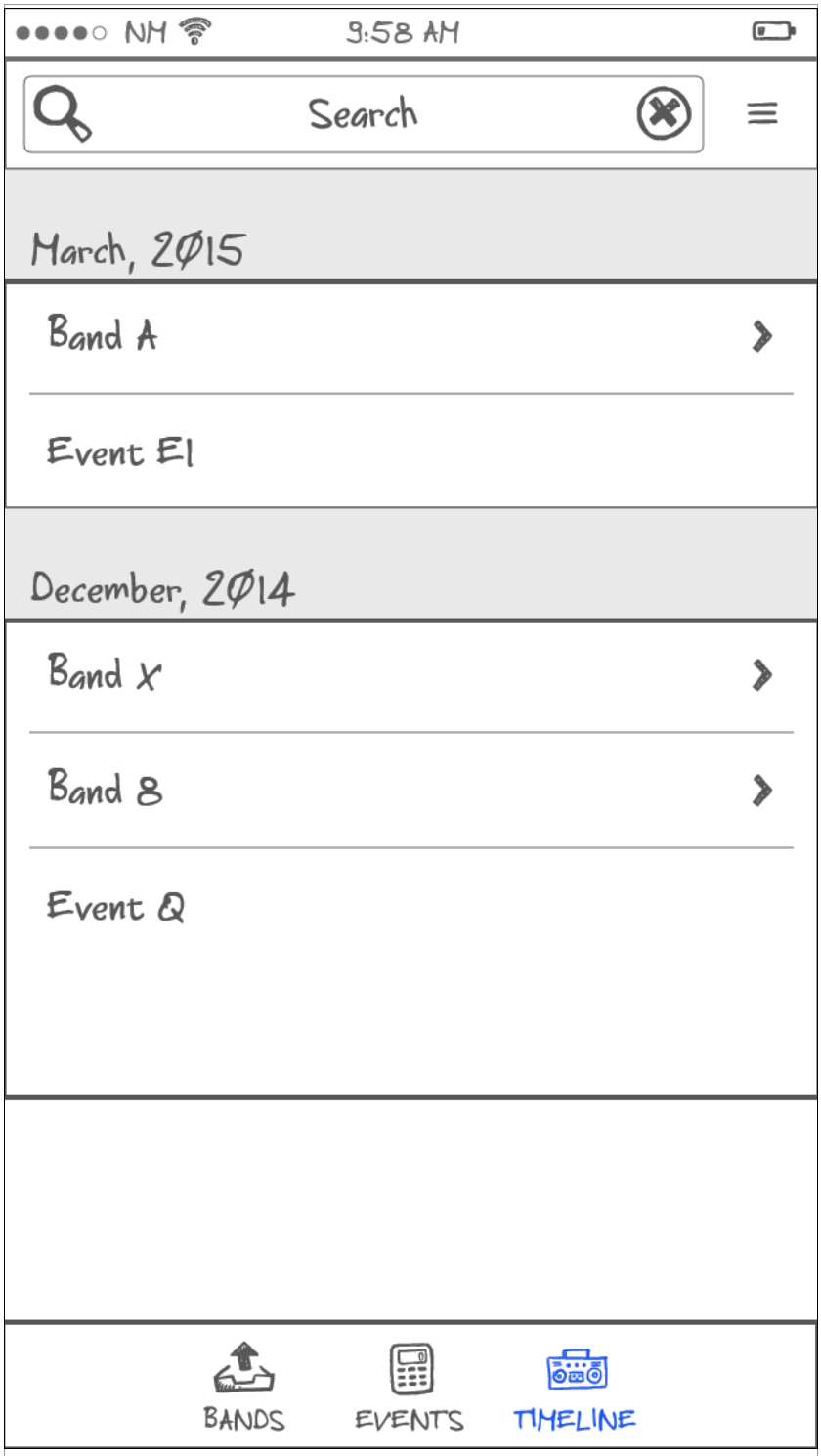
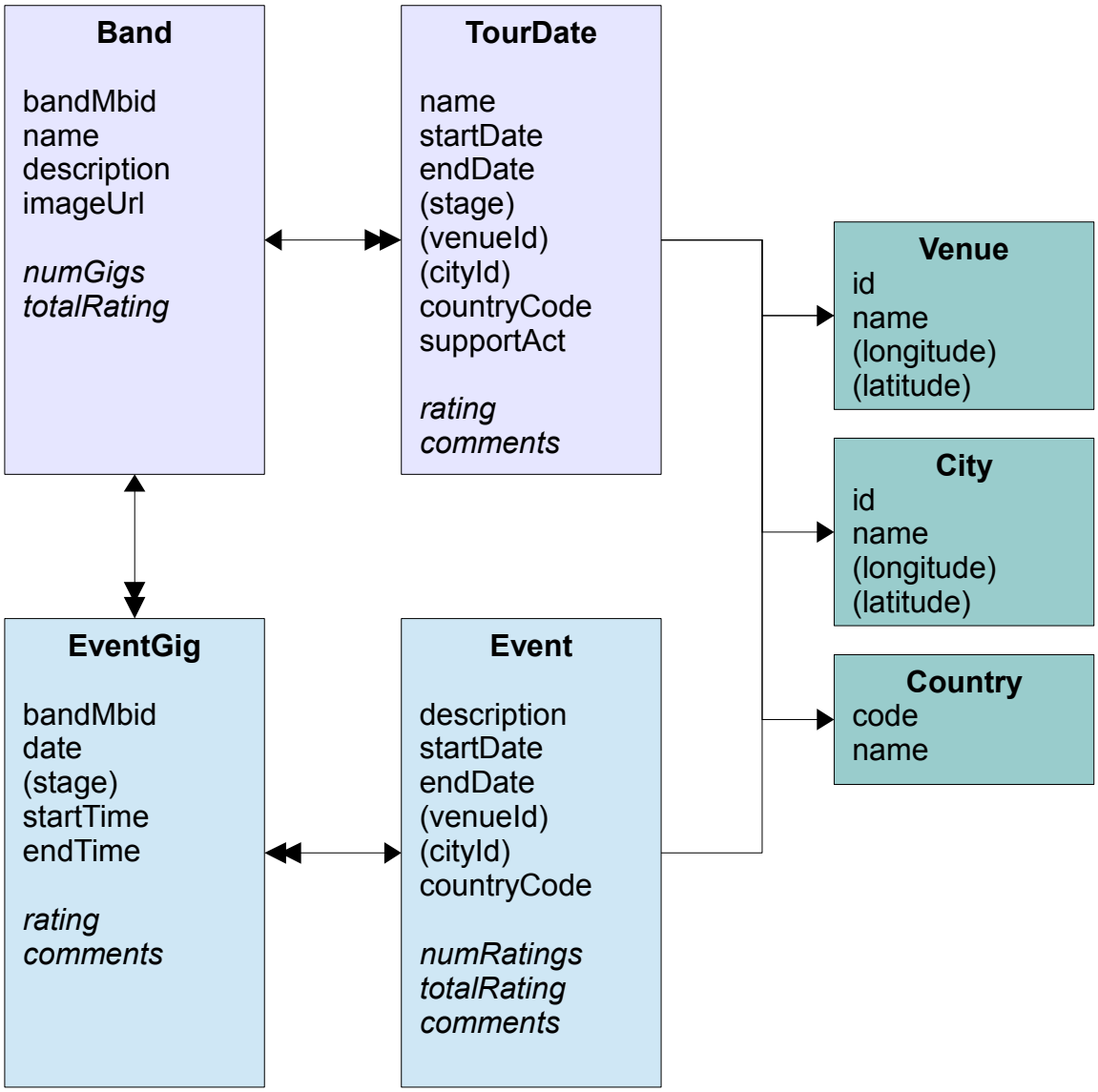


Illustration 13: Timeline

4.2. Local database



4.3. Implementation details

4.3.1 Accessing setlist.fm

See http://api.setlist.fm for more information. Use the http://api.setlist.fm/rest/0.1/search/setlists endpoint with artistMbid and date parameters and parse the results.

4.3.2 Accessing Youtube

See https://developers.google.com/youtube/v3/docs/search/list for more information.

5. Development cycle and milestones

Server		Implement API's to serve information about artists from manually created dataset. (see chapter 3.3.1)
Client		Search for artists on the server and add them to local database (see chapter 4.1.1 and 4.1.2)
Milestone-1		List of artists (no gigs, ratings)
Server		Add token-based authentication to all request
Client		Add token-based authentication to all request
Milestone-2		Authenticated requests
Server		Implement API's to serve information about venues, cities and countries.
Client		Manually create gigs
Milestone-3		Basic functionality in bands view (add bands / gigs)
Client		Add set-list and YouTube videos to gig detail
Milestone-4		Full functionality in bands detail view
Server		Implement API's to serve information about tourdates
Client		Add guided creation of gigs
Milestone-5		Full functionality in bands detail view
Server		Implement API's to serve information about events
Client		Add guided creation of events
Milestone-6		Basic functionality of events view
Client		Add manual creation of events
Client		Add timeline view
Milestone-7		Complete functionality for submission to Udacity
Client		Polish general UX – UI for rotated views
CodeReview		Submission to Udacity

Potential extra features before submission to Apple:

- tablet specific lay-outs
- social features (see ratings etc. of facebook friends)

6. Appendix – external resources

6.1. Development tools

XCode	For development of the iOS app	-
Visual Studio 2013	For development of the server component.	Community Edition
NinjaMock	For the screen mockups	Non-commercial use.
OpenOffice	Documentation	Apache License 2.0

6.2. Server

Node.JS	JavaScript runtime	ICU License
Express	Web framework for Node.js	MIT License
MongoDB	Database backend	GNU AGPL
Mongoose	MongoDB object modeling for Node.js	MIT License
Cheerio	core jQuery implementation for Node.js	MIT License

6.3. IOS App

SwiftyJSON	JSON parser for Swift	MIT License