TEXAS LAW DATABASE

Team: For-git about it

Members: Jimmy Kettler, Eliya Shabanov, Ian Hays, Keyon Mohebzad, Paul Strong, Michael Dong

Contents

Introduction

Problem

The democratic process works best if there is a well-informed citizenry. Given more information, voters can make more informed decisions on their support for political candidates. However, there currently does not exist a well-made central database containing information concerning the Texas Senate. There is no place to see comprehensive information on bills, senators, committees and the relationship between them. Without such a database, it is difficult for citizens to hold their government accountable. The ability for voters to see a candidate's voting record allows them to decide for themselves whether a candidate is suitable for the job. While such databases exist for the federal government (https://www.govtrack.us/ is an example), the only place where one can access a database for bills going through the Texas legislature is through the official Texas legislature website, which is lacking in information. The Texas Law Database exists to provide such a service, to open source democracy in Texas, and to serve as a source of information so that voters can learn about bills, legislators, and committees in order to stay informed about the democratic process in Texas.

Use Cases

The Texas Law Database contains the following things:

Information about specific bills that are going through, or have gone through, the Texas Senate, including a brief summary, author, status, a link to the full text of the bill, the committee that the bill went through, and a breakdown of the votes for the bill.

Information about individual senators, including a brief biography, their voting history, a list of committees they chair or are a member of, and a link to their facebook and/or twitter, if they have one.

Information about Senate committees, including a description, their members, and a list of all bills to have gone through that committee.

Users of the Texas Law Database can learn about the most recent bills to go through committee or a floor vote, or they can look through the lists of senators and committees and see information about the bills they have voted on.

Users can use the Texas Law Database to see Senator's voting records over time, and thus get a better understanding of a senator's position on issues. Voters can then use this information to affect their voting decision.

Users can see a list of bills that passed or failed and gain a greater understanding of the legal trends for the Senate as a whole. Users in the political field can use this information to make informed predictions on proposed bills or bills to be proposed.

The net effect of this information is to allow voters in Texas to contribute to and to learn about the Texas senate, to inject some much needed openness in the democratic process, and to give voters and citizens in Texas the tools to make informed decisions about their Senate.

Design

Design Overview

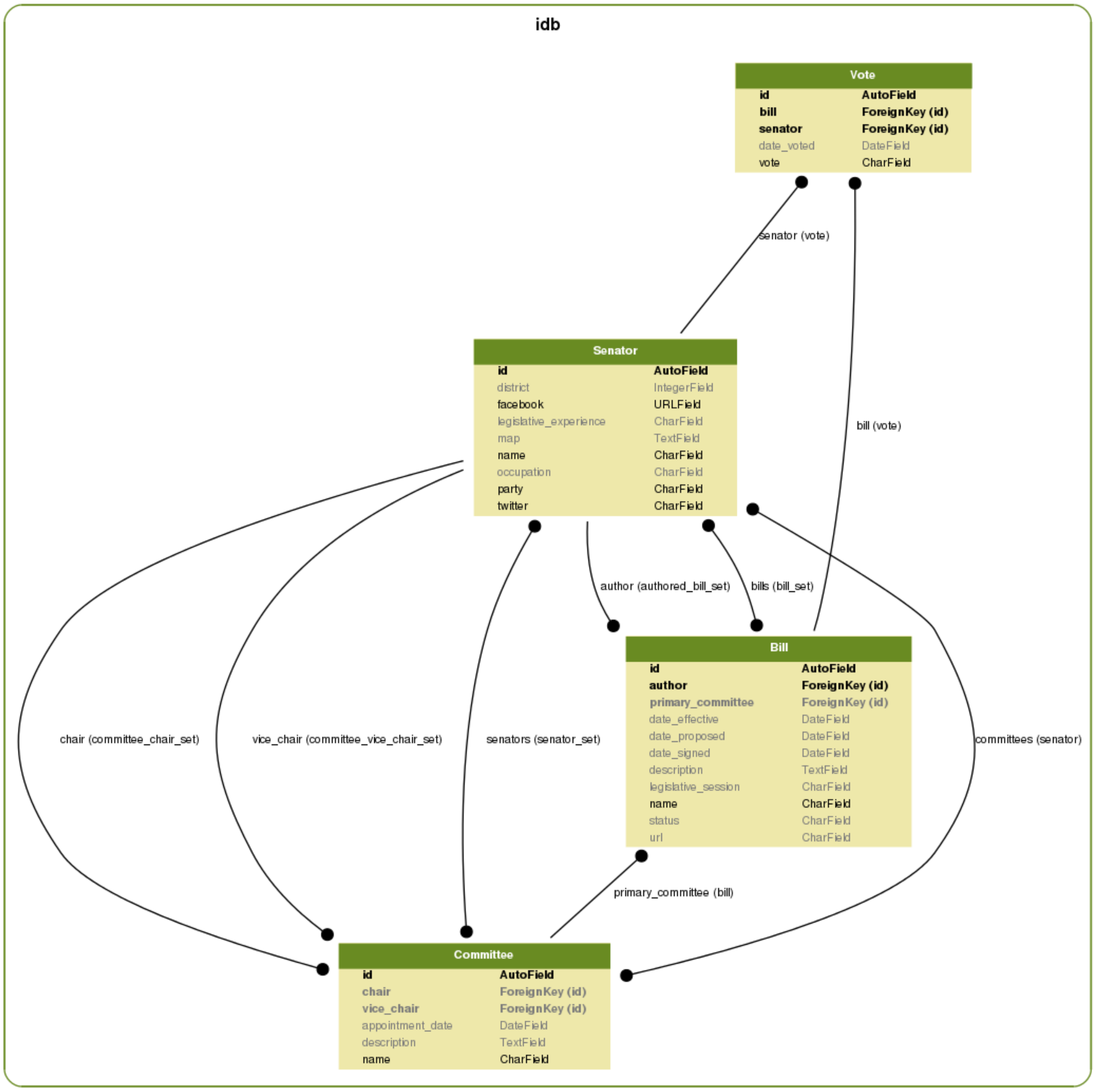
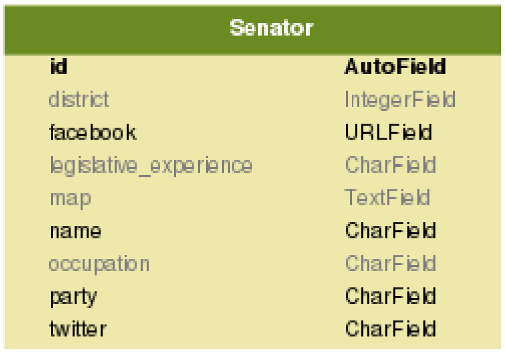


Figure 1: UML Overview of Models

Django Models

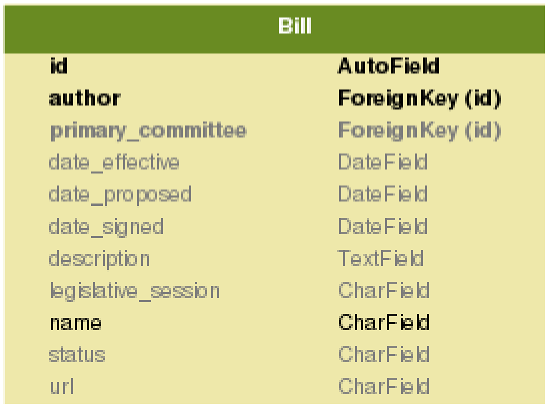
Senator Model



Senator	
id	**AutoField**
district	IntegerField
facebook	URLField
legislative_experience	CharField
map	TextField
name	CharField
occupation	CharField
party	CharField
twitter	CharField

The senator model models each of the 31 individual senators in the Texas Senate. These are the legislators in the Texas Senate who author, sponsor, and vote on bills, whether in committees or on the floor. The primary key is an auto-generated integer ID supplied by Django. Attributes include their name, district, party, occupation, legislative experience, and their facebook and twitter pages if they have one.

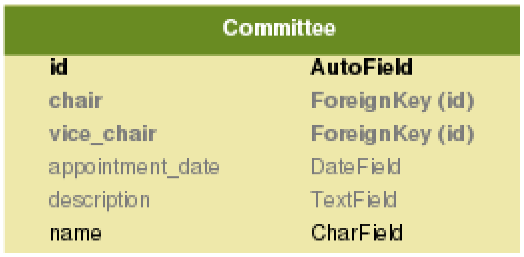
Bill Model



Bill	
id	**AutoField**
author	**ForeignKey (id)**
primary_committee	**ForeignKey (id)**
date_effective	DateField
date_proposed	DateField
date_signed	DateField
description	TextField
legislative_session	CharField
name	CharField
status	CharField
url	CharField

The bill model models specific bills submitted to committees or the senate floor at large. These bills are bills that have gone to a vote in committees or on the floor. As with senators, their primary key is an auto-generated integer supplied by Django. Attributes include the name of the bill, the legislative session in which the bill was proposed, the date it was proposed, and if passed, the date it was signed into law and the date it was effective. Attributes also include its status, a link to its full text, and a brief description on the contents of the bill.

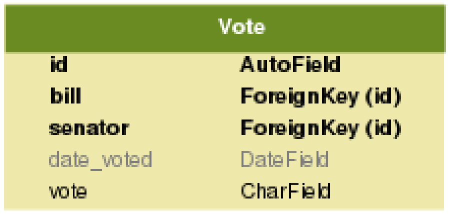
Committee Model



Committee	
id	**AutoField**
chair	**ForeignKey (id)**
vice_chair	**ForeignKey (id)**
appointment_date	DateField
description	TextField
name	CharField

The committee model models the 31 standing committees in the Texas Senate. Every bill must go through committee, where it is debated, amended, and voted on before it goes to the floor where it is voted on by the Senate as a whole. As with senators and bills, the primary key is an auto-generated integer supplied by Django. Attributes include its name, a description, and the date that the committee was first formed.

Vote Model



Vote	
id	**AutoField**
bill	**ForeignKey (id)**
senator	**ForeignKey (id)**
date_voted	DateField
vote	CharField

The vote model models the votes that senators cast for bills. Senators may vote Aye or Nay, or they may be present but not voting, or absent from the vote. We store the actual value of the vote itself as a tuple of strings, a 3 letter abbreviation for the value of the vote, and its full English value. We also store as an attribute the date of the vote.

Senator and Bill Relationship

Bills are authored by Senators, and although there are co-authors, we have chosen to only include the main author, for simplicity's sake. A bill can have a long list of co-authors, some of which may not be senators, and thus, as an initial step, we have decided only to include main authors. Senators can author multiple bills. Therefore, we have a many to one relationship between bills and senators.

Senator and Committee Relationship

There are 31 standing committees in the Texas Senate. Each committee is chaired by a senator and vice chaired by a senator, though those positions may be vacant at any given time. Each committee also has some number of senators as members, usually less than 10. A senator may be part of any number of committees. Therefore, there exists a many to many relationship between senators and committees. Since there are multiple types of relationship between senators and committees: members, vice chairs, and chairs, a senator's membership of a committee is modeled by the three different sets a senator to committee relation can belong to: committee_chair_set, committee_vice_chair_set, and committee_senators_set.

Bill and Committee Relationship

Each bill must go through a committee. While a bill may go through multiple committees before going to the floor for a vote, and indeed, every bill must go through the calendar committee to be placed on the floor for a vote, each bill mainly falls under the purview of one committee. This is the committee which we have chosen to model. Each committee oversees many bills. Therefore, there exists a many to one relationship between bills and committees.

RESTful API

JSON

Our API is a RESTful API that performs transactions using JSON, a lightweight standard format that uses human readable text to transmit data objects consisting of attribute-value pairs. Although originally derived from JavaScript, its simplicity and power make it an extremely widespread web standard. We separated our HTML returns and our JSON returns by prepending /api. Example: API endpoint for '/people' would be '/api/people/' .

GET

Example snippets of JSON requests and replies are shown below for the Senator class. Similar requests and replies are analogous for the other classes.

/api/senators - GET

Gets all of the senators. Example:

```
+ Response 200 (application/json)

        [
            {
                "id": 1,
                "name": "Jane Nelson",
                "party": "Republican",
                "occupation": "Businesswoman, former teacher",
                "legistlative_experience": "Disaster Relief",
                "district": "12",
                "twitter": "https://twitter.com/SenJaneNelson",
                "facebook": "https://www.facebook.com/SenatorJaneNelson",
                "picture": "none",
                "committees": [1,2]
            },
            {
                "id": 2,
                "name": "John Whitmire",
                "party": "Democratic",
                "occupation": "Attorney",
                "legistlative_experience": "",
                "district": "15",
                "twitter": "",
                "facebook": "",
                "picture": "none",
                "committees": [2]
            }
        ]
```

/api/senators/{id} - GET

Gets one of the senators. Example:

```
+ Response 200 (application/json)
    + Body

            {
                "id": 1,
                "name": "Jane Nelson",
                "party": "Republican",
                "occupation": "Businesswoman, former teacher",
                "legistlative_experience": "Disaster Relief",
                "district": "12",
                "twitter": "https://twitter.com/SenJaneNelson",
                "facebook": "https://www.facebook.com/SenatorJaneNelson",
                "picture": "none",
                "committees": [1,2]
            }
```

/api/senators/{id}/bills - GET

```
All of the Bills that this senator has authored
###List the bills [GET]
+ Response 200 (application/json)

        [
            {
                "id": 1,
                "name": "SB 63",
                "author": [1],
                "legislative_session": "83(R)",
                "date_proposed": "11/12/2012",
                "date_signed": "6/14/2013",
                "date_effective": "6/14/2013",
                "status": "Signed into law",
                "url":
"http://www.legis.state.tx.us/BillLookup/History.aspx?LegSess=83R&Bill=SB63",
                "primary_committee": 1,
                "Description": "Relating to consent to the immunization of certain children.",
                "votes": {"1": "AYE","2": "NAY"}
            }
        ]
```

Others

- /api/senators/{id}/bills – GET – Gets all of the bills that the senator with id {id} has authored
- /api/bills – GET – Gets a collection of all of the bills in the database
- /api/bills/{id} – GET – Gets a specific bill, where {id} is the identification key of the desired bill
- api/bills/{id}/senators – GET – Gets a list of senators and their corresponding votes for the bill with id {id}
- api/bills/{id}/authors – GET – Gets a list of the senators who authored the bill with id {id}
- /api/committees – GET – Gets a collection of all of the committees in the database
- /api/committees/{id} – GET – Gets a specific committee, where {id} is the identification key of the desired committee
- /api/committees/{id}/senators – GET – Get all the senators in the committee with id {id}
- /api/committees/{id}/bills – GET – Get all the bills originating in the committee with id {id}

PUT

/api/senators/{id} - PUSH

Update an existing senator

```
+ Request (application/json)

        {
            "id": 1,
            "name": "Jane Nelson",
            "party": "Republican",
            "occupation": "Businesswoman, former teacher",
            "legistlative_experience": "Disaster Relief",
            "district": "12",
            "twitter": "https://twitter.com/SenJaneNelson",
            "facebook": "https://www.facebook.com/SenatorJaneNelson",
            "picture": "none",
            "committees": [1,2]
        }
```

```
+ Response 204
```

Others

- /api/bills/{id} – PUT – Updates a specific bill, where {id} is the identification key of the desired bill
- /api/committees/{id} – PUT – Updates a specific committee, where {id} is the identification key of the desired committee

POST

/api/senators/{id} - POST

Create a new senator entry

```
+ Request (application/json)

        [
            {
                "name": "Jane Nelson",
                "party": "Republican",
                "occupation": "Businesswoman, former teacher",
                "legistlative_experience": "Disaster Relief",
                "district": "12",
                "twitter": "https://twitter.com/SenJaneNelson",
                "facebook": "https://www.facebook.com/SenatorJaneNelson",
                "picture": "none",
                "committees": [1,2]
            }
        ]

+ Response 201 (application/json)

        {
            "id": 1
        }
```

Others

- /api/bills – POST – Adds a new bill to the collection of bills
- /api/committees – POST – Adds a new committee to the collection of committees

DELETE

/api/senators/{id} – DELETE

```
+ Response 204
```

Others

- /api/bills/{id} – DELETE – Delete a bill from the database, where {id} is the identification key of the bill to remove
- /api/committees/{id} – DELETE – Delete a committee from the database, where {id} is the identification key of the committee to remove

API Unit Tests

Unit tests for the API, found in cs373/idb/tests.py, use the Python unittest library to individually test each endpoint and the associated HTTP methods. At the start of each test method we open a connection to the Apiary-mock servers which we close at the end of the method. In each test, the response status of the HTTP request is checked. In the methods that return a response body, the body is in byte form and must be decoded to a string and then loaded into JSON using the “json.loads” function.

In the tests for the POST and PUT HTTP methods, the variable “values” is used with the “json.dumps” function to convert a python list/dictionary into a JSON object to be passed to the client. For the POST method, the response body is a JSON object containing only the id of the new object. For the PUT method, there is no response body check.

In the tests for the GET HTTP methods, the variable “desired_body” contains the expected response body in the form of a python object that is checked against the response returned from the client.

In the tests for the DELETE HTTP methods, there is no “desired_body” or “values” variables as no JSON objects are passed to or from the client.