Computer Engineering Department

Enterprise Backend as a Service - *A code that writes code*

Project Advisor: Prof. Gokay Saldamli

Bodiwala, Maulin(MS Computer Engineering)
Doshatti, Aditya (MS Software Engineering)
Kapadia, Darshil (MS Software Engineering)
Nyati, Devashish (MS Software Engineering)

Introduction

Web Application is nothing but a piece of software running on a remote computer which can then be accessed by anyone and at anytime over the internet.The development of web applications would not have been this popular if the concept of RESTful(REpresentational State Transfer) system had not come into the picture

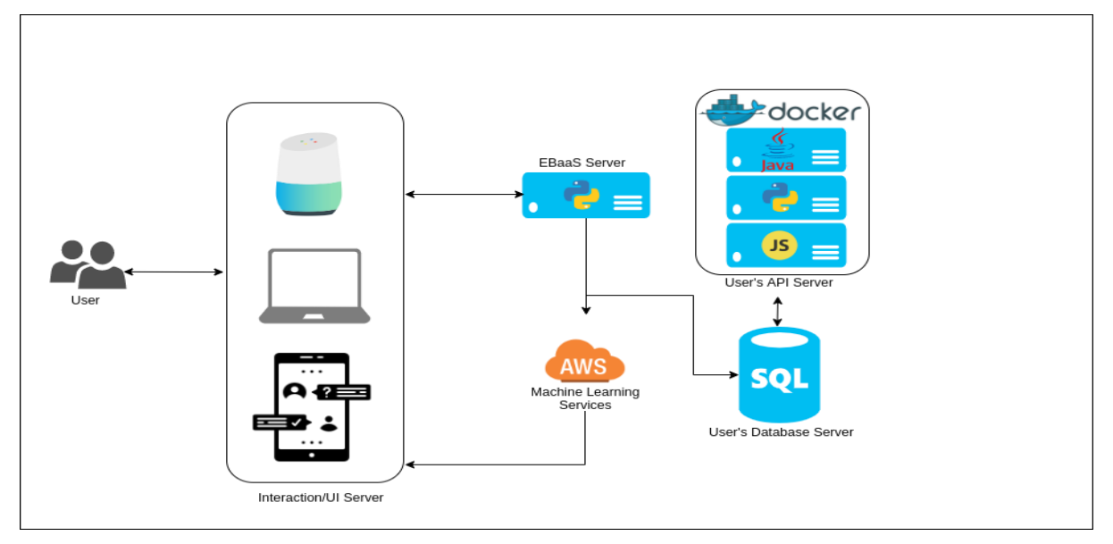
In the world of startups, the need for fast development of applications is rising. Companies constantly urge developers to create applications and other business software more quickly without sacrificing quality. In such situation, having a solution which provides RESTful APIs on the go based on the database details can be seen as a blessing.

To address the situation, our work provides a solution by offering enterprise backend as a service.Our application communicates with the user to get their requirements in terms of database and takes care of the tasks like creating the database, hosting the database and creating a CRUD based RESTful APIs and providing the code for the same.

Automated generation of back-end API's can save a lot of time. It can benefit not only IT experts, non IT people, but also small organizations or start ups.

Problem Statement

Our problem statement is to build an application that will interact with the user to create a backend code that should create the required product with bare minimum requirements. Starting from the creation of the database to the creation of the APIs to communicate with the database, the user should be able to build applications in a few minutes with just a few clicks. The user should be able to achieve 3 main goals:



Build Application:

The user should build the backend applications with just a few clicks and be able to host them on different cloud services.

Generate Code:

The user should be able to generate the backend code without having any prior knowledge of coding.

Build Databases:

The user should be able to migrate their existing databases or create new databases easily using our application.

Methodology

The project aims for allowing user to control the information from nothing but just the mere knowledge of the user of what information he/she wants to manage. Putting it in a technical context, the project is trying to provide the user with a backend that gives control to the models which are nothing but a representation of the information the user wanted to manage. Breaking it into various processes, two major processes stand out:

Generating Models

The application generates the models based on the user requirements which is then mapped to a database entity. The application allows user not just to connect to an existing database but also allows to create one from scratch.

After the creation or connecting to a existing one, the user can move ahead with modifying the database by adding/removing tables, columns, and relationships. Based on their operations, the corresponding changes will be made to the database. Behind the scenes, the application is focusing on operating and manipulating database models through simple SQL queries known to the whole world in a systematic and a more structural way. The progressive way of asking the information from the user leads to successful generation of models.

Generating Code

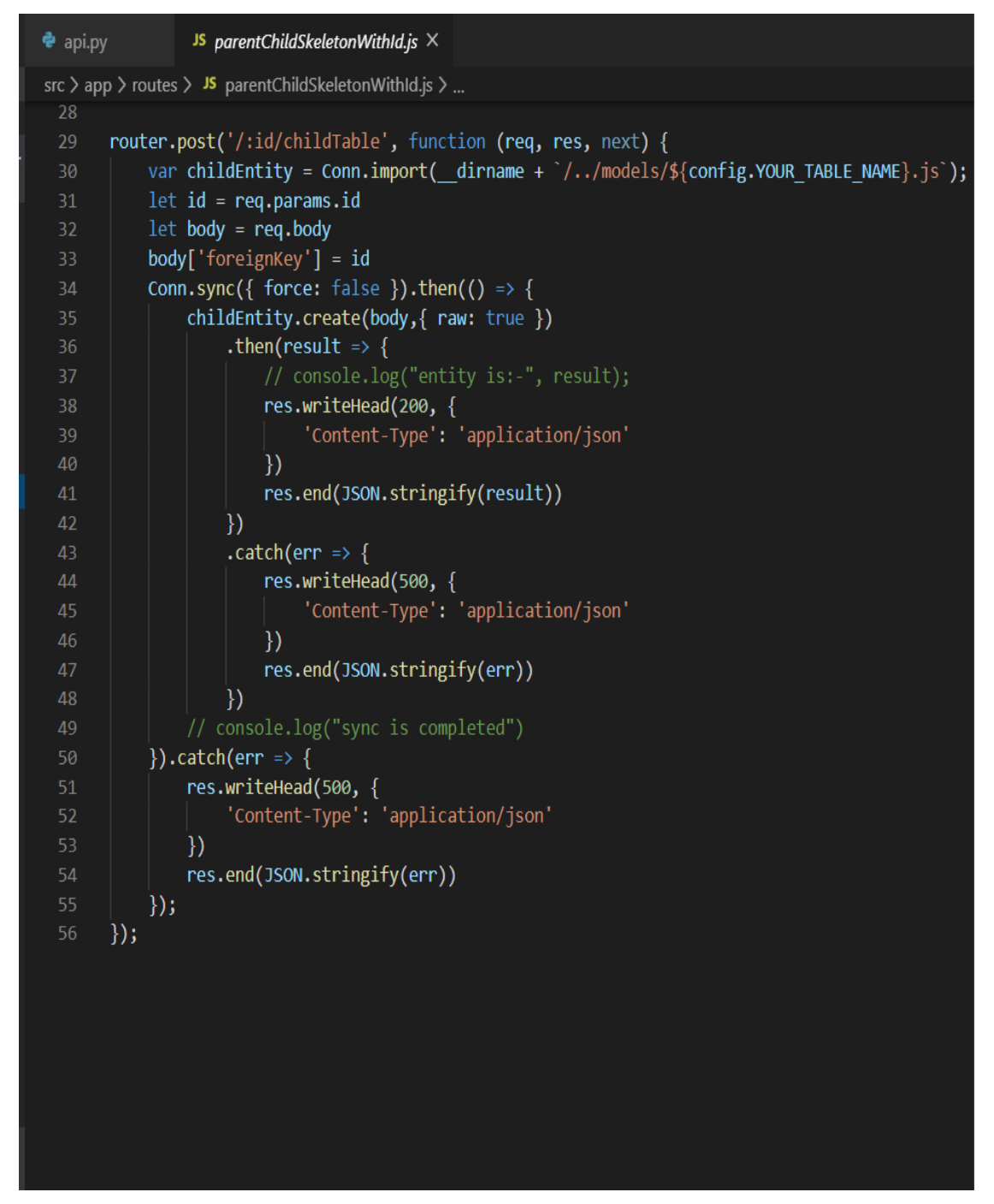
The generated code uses the REST Architecture for exploiting the advantages it has over SOAP, GraphQL and others. JavaScript as the language for backend considering its popularity. Once that was decided, the generation of code involved various processes as follows:

1) Mapping Models: This process indulges in converting the database models to JavaScript Objects. We used Sequelize ORM to exploit its features for generating the Javascript entities directly from the provided database schema. Sequelize also takes care of mapping the relations between models.

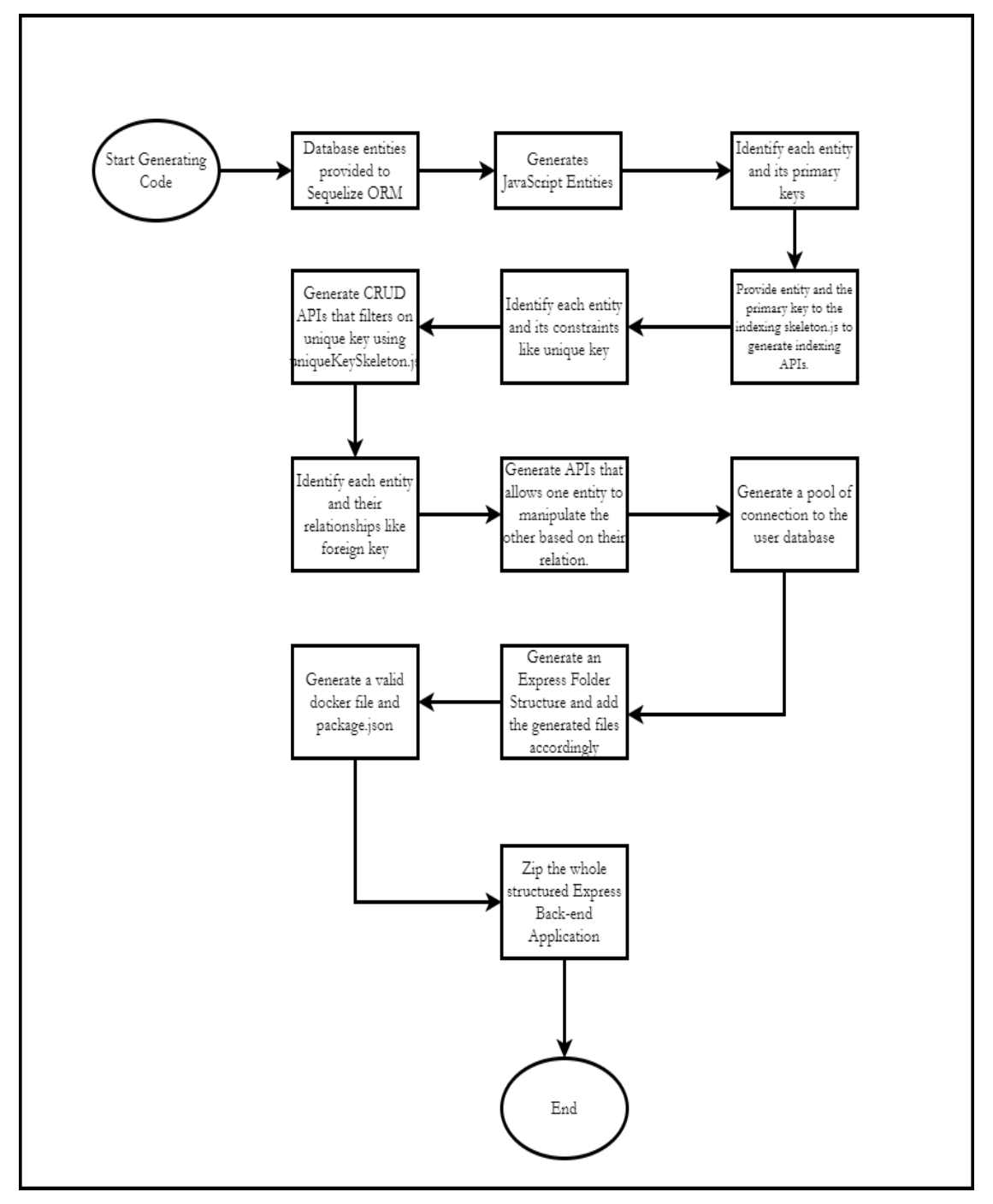
2) File Structure: We decided to organize and present the files around features and not roles to bring the uniformity across the back-end code. Route files separated from "index.js" which works as the router. This file structure ensures the best practices are followed and brings modularity in terms of structure.

3) Database Connection: Coming to the connection to the database, the project manages a pool of connection which eliminates problems related to connections like open connections.

4) URIs: Which URIs to provide was the most critical part. The project only kept focus on generating CRUD operations APIs through skeletons as shown below. To not overcrowd the users with APIs, we decided to project APIs based on :


api.py
parentChildSkeletonWithId.js
src > app > routes > parentChildSkeletonWithId.js > ...

```javascript
router.post('/:id/childTable', function (req, res, next) {
    var childEntity = Conn.import(__dirname + `/../models/${config.YOUR_TABLE_NAME}.js`);
    let id = req.params.id
    let body = req.body
    body['foreignKey'] = id
    Conn.sync({ force: false }).then(() => {
        childEntity.create(body,{ raw: true })
            .then(result => {
                // console.log("entity is:-", result);
                res.writeHead(200, {
                    'Content-Type': 'application/json'
                })
                res.end(JSON.stringify(result))
            })
            .catch(err => {
                res.writeHead(500, {
                    'Content-Type': 'application/json'
                })
                res.end(JSON.stringify(err))
            })
        // console.log("sync is completed")
    }).catch(err => {
        res.writeHead(500, {
            'Content-Type': 'application/json'
        })
        res.end(JSON.stringify(err))
    });
});
```


Start Generating Code
Database entities provided to Sequelize ORM
Generates JavaScript Entities
Identify each entity and its primary keys
Provide entity and the primary key to the indexing skeleton.js to generate indexing APIs.
Identify each entity and its constraints like unique key
Generate CRUD APIs that filters on unique key using uniqueKeySkeleton.js
Identify each entity and their relationships like foreign key
Generate APIs that allows one entity to manipulate the other based on their relation.
Generate a pool of connection to the user database
Generate an Express Folder Structure and add the generated files accordingly
Generate a valid docker file and package.json
Zip the whole structured Express Back-end Application
End

i) Primary keys: APIs that uses primary keys for indexing.
ii) Constraints: APIs that uses constraints like unique key to manipulate record based on that field.
iii) Relationships: APIs that control one entity using the relation it has with the other entity. E.g.: Foreign Key.

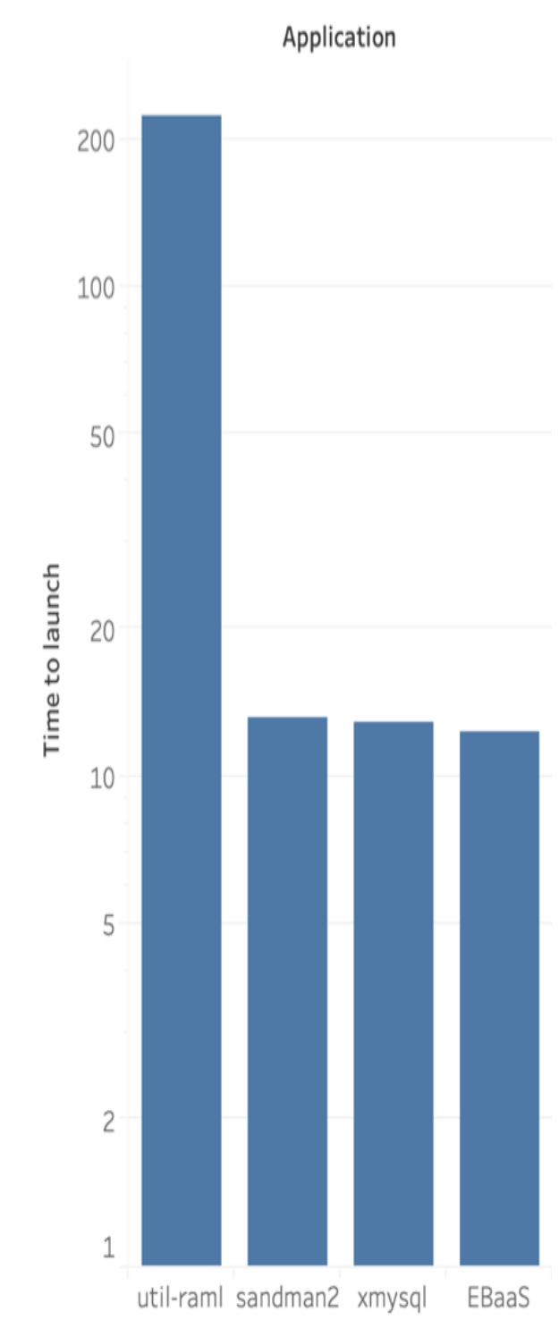
Analysis and Results

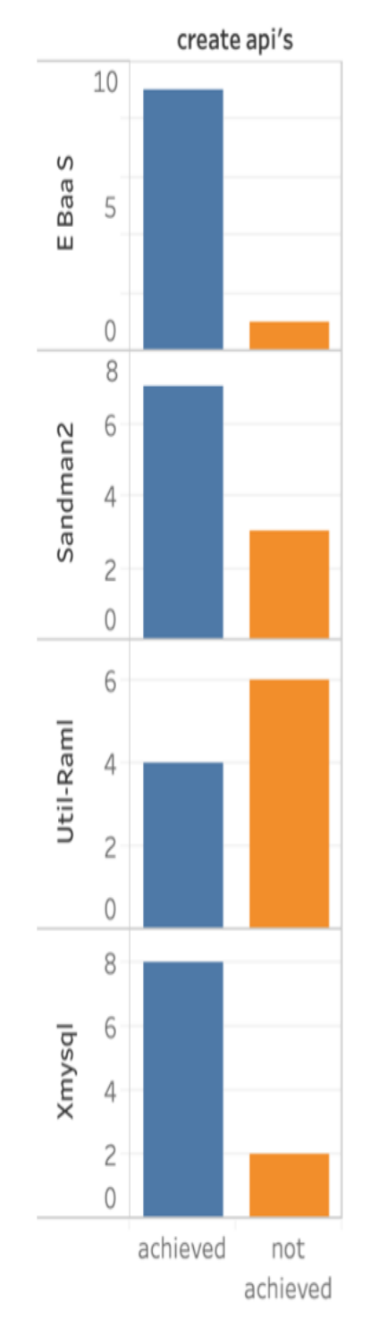
The application was evaluated by back-end developers, data scientists, and non technical people.

	util-raml	xmysql	sandman2	EBaaS
Usability	2/5	3/5	3/5	5/5
Speediness	2/5	5/5	5/5	4/5
Efficiency	3/5	2/5	2/5	4/5
Code Quality	4/5	N/A	N/A	4/5
Database Connectivity	N/A	N/A	4/5	4/5
Cloud Services	N/A	4/5	4/5	5/5

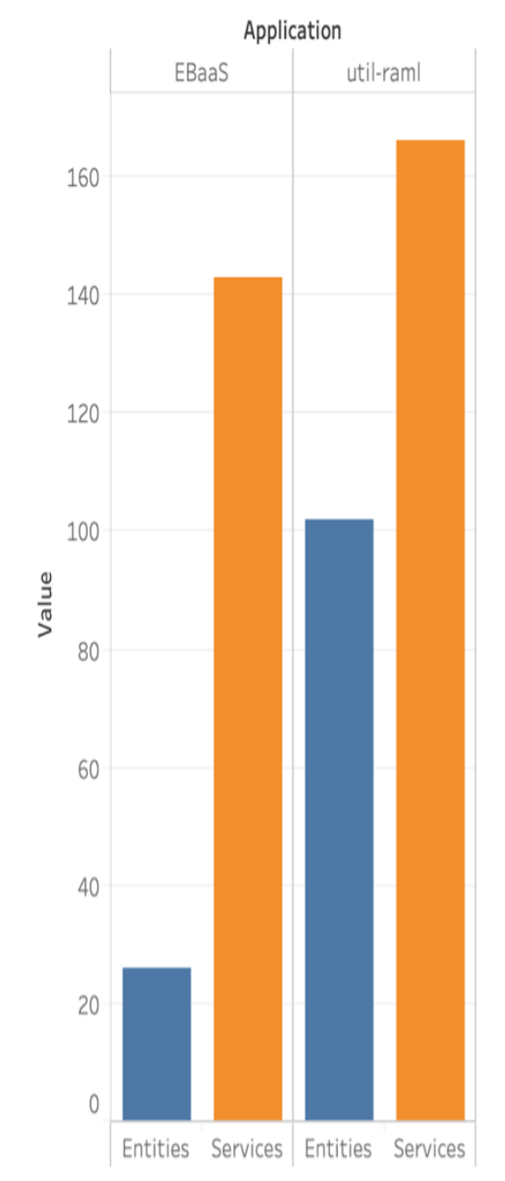
The users evaluated the applications as follows:

- ***Usability:*** EBaaS performed extremely well as compared to other code generators. 9 out of 10 users were successfully able to launch the applications on EBaaS. Since users don't need any prior coding knowledge to launch an application, it was very easy for users from non technical background also.
- ***Speediness:*** The average time to connect to a database and launch an application on EBaaS is around 12 seconds. Other code generators took upto 4 minutes.





- ***Code Quality:*** EBaaS gives the JavaScript code that can be easily modified and managed later. The number of lines of codes were comparatively less as compared to other code generators.



- ***Database Connectivity:*** EBaaS gives the best database connectivity as compared to other code generators. Users can connect to existing databases or create new databases from sql files or excel using the EBaaS interface. Users can also modify the tables and relationships in any existing database.
- ***Cloud Services:*** The world is quickly moving towards cloud computing which allows software to be available based on demand. To make it even more feasible, containerization has taken its own place. Keeping this in mind, EBaaS provides a docker file on hand which provides users to easily make their backend application containerized. Util-raml-code-generator fails to do so.

Summary

EBaaS is a one stop application for developing and maintaining the complete back end of any application. Interacting with the UI, user can develop database, make updates in the database and have the API code ready to integrate the APIs with their own front end. User with their own servers can have the complete, efficient and speedy back end with all basic APIs ready with just few clicks and inputs. User can later make any changes in the API code provided by our code generator to also have user defined APIs. So EBaaS basically, as the name suggested, is an application to provide enterprise backed as a service.

Key References

[1] Crocombe, R., & Kolovos, D. S. (2015, September). Code Generation as a Service. In CloudMDE@ MoDELS (pp. 25-30).

[2] Zenqry.com. (2019). ZenQuery - Enterprise Backend as a Service.

[3] GitHub Paysera. (2019). paysera/util-raml-code-generator.

[4] xmysql codebase: https://github.com/o1lab/xmysql

[5] X. Qafmolla and V. C. Nguyen. Automation of Web Services Development Using Model Driven Techniques. In ICCAE conf, volume 3, pages 190–194, 2010.

[6] Hamza Ed-douibi, Javier Luis Cánovas Izquierdo, Abel Gómez, Massimo Tisi, and Jordi Cabot. 2016. EMF-REST: generation of RESTful APIs from models. In Proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC '16).

Acknowledgments

We would like to express our gratitude for all the help that our project advisor Prof. Gokay Saldamli has provided throughout the duration of our project "Enterprise Backend as a Service". Professor was always there to guide us through every difficulty we faced on our path. Apart from that, we sincerely thank Prof. Liu and Prof. Harkey for updating us with every small details related to project. They were always available to provide support for any matter we took to them.

Last but not the least, we thank San Jose State University for allowing us to use the resources, without which this would have not been possible.