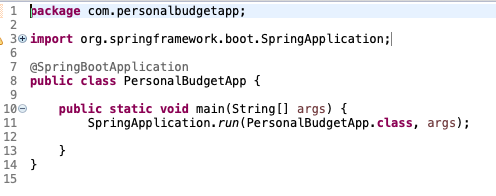
Personal Budgeting Application

Manual and Documentation

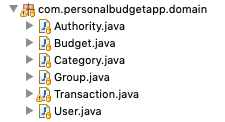
By Liam Heywood

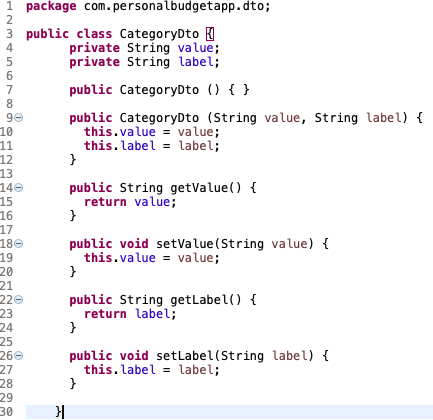
**Contents**

1. Back-End File Setup
   1. – Driver
   2. – Domain
   3. – DTO
   4. – Repository
   5. – Security
   6. – Service
   7. – Util
   8. – Web
2. Front-End File Setup
   1. – HTML/CSS/JS
3. Using the application
4. Future developments
   1. Driver

The driver is the entry point of the application. As you can see there are only 14 lines of code. Notice on line 3 this is where the application imports the Spring framework and allows us to actually run the application.

* 1. Domain

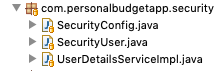
The domain package is where we configure all of our MySQL database tables. We initially began with just four but as I began developing I realized it would be useful to have an authority and budget table. If you take a closer look at the code (available at <https://github.com/lheywood/BudgetApp> ), you can see I set primary and foreign keys, and basically mapped out how the data is related.

* 1. Data Transfer Object

Here the DTO class has only one purpose. To take data in and then output it in a form that benefits the application. As you can see there are only getters, setters and a constructor. What this does is when we are creating a transaction, when we begin typing in the input box for category it searches for all categories that are currently in the user’s category table so we don’t have to type the whole category name.

* 1. Repository

The repositories do not require much code at all, they extend a JpaRepository from the Spring framework that gives us generic CRUD methods from our database. Basically, this package communicates with our database and has the power to retrieve and pass data to our tables.

* 1. Security

Security was being very important, especially when we are dealing with people transactions and finances. Luckily, Spring provides a very useful system when it comes to creating user accounts and password security.

The SecurityConfig is where we grant permission to a given user to certain pages on our front end. In this case if you were to take a look at the code you would see that all CSS/JS are accessible to all as well as the login and register page. However, you must have an account in our database a USER role in order to access the budgets page. If a user without an account tried to access the budgets endpoint they will be redirected to the login page.

The SecurityUser is where we basically set up the current user, we get then set their id, username, password, budgets so that when the hit the budgets endpoint only their data is displayed.

UserDetailsServiceImpl is where we implement Spring’s UserDetailsService. This allows us to search for the current user’s data. In our case we create a custom method that finds the users data based on the username. Notice we also use the UserRepository to access our data.

* 1. Service

The service package uses the respective repository to manipulate the user’s data and acts receives a message from the front end which it then utilizes the repository to then actually manipulate the data and carry out the message.

* 1. Util



A very simple package, this has only two methods but was required so that our application and database were working with the same date format so we could allow the user to change budget time frames.

* 1. Web

In the web package we have our controllers. The role of each controller is to manage the flow of data between entities. In this case it controls the data between the front and our model. When a user adds a transaction for example the controller adds the new data onto the model so that it can be accessed elsewhere and added to the database.

1. HTML/CSS/JS

The front end was made up of simple HTML files. The important aspects of the application were the back-end features therefore to get the application up and running the front end was not a major concern. I used bootstrap to make the site a little bit more professional in terms of design. Also, it allowed the application to look almost identical on a mobile device which is the main target device.

Thymeleaf provided a way to populate the endpoints with data from our backend without complicating things in regards to our HTML elements.

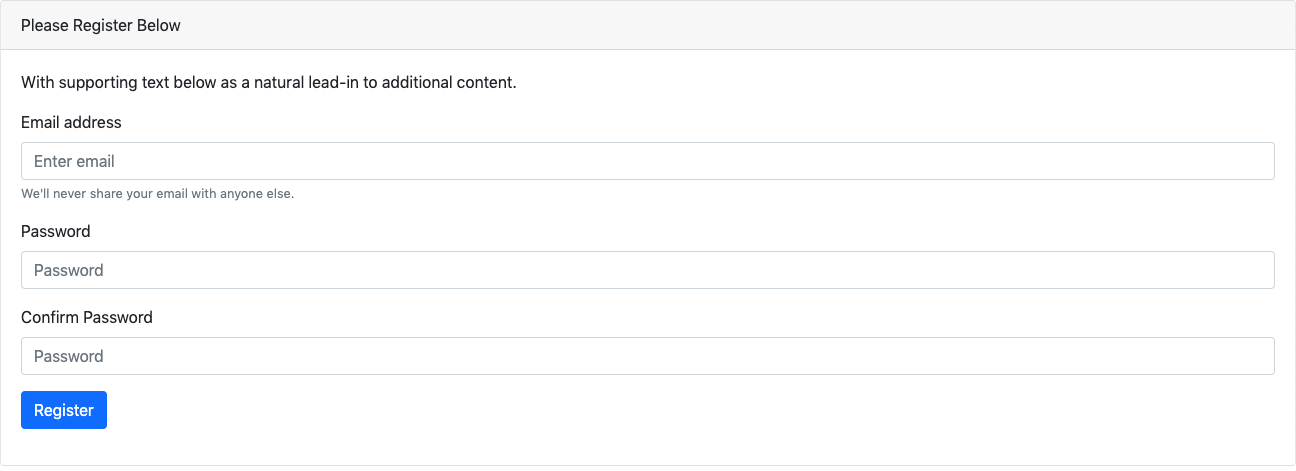
There are no major CSS files but were included for future development purposes. However, we did use inline styling in some cases just because bootstrap didn’t format everything as I expected.

JavaScript was used to display error messages if we failed to login, form submissions and also to change the date range on the budget endpoint.

1. Using the Application

This is a tutorial on how a new user would begin using the application. This tutorial begins at the registration page as it assumes the user understands the fact they need to click register on the login page.

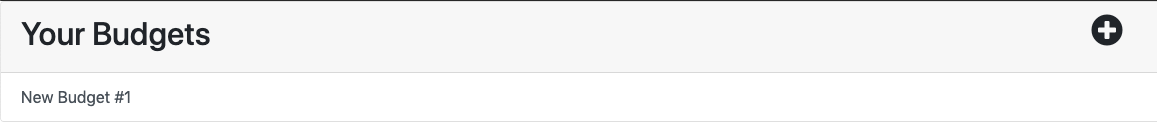
**Step 1: Register**



The user inputs a valid email address and password twice. The email input automatically checks to see if the input could be a valid email address like somestring@anotherstring.com.

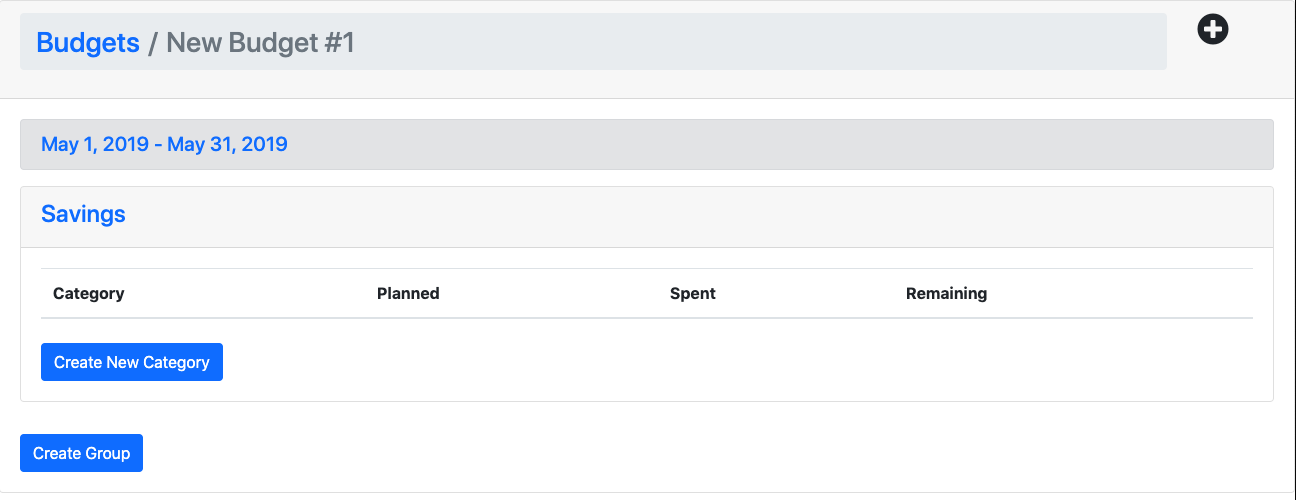
The password fields perform a check against one another and if false sends a message and will not create an account.

**Step 2: Create a budget**



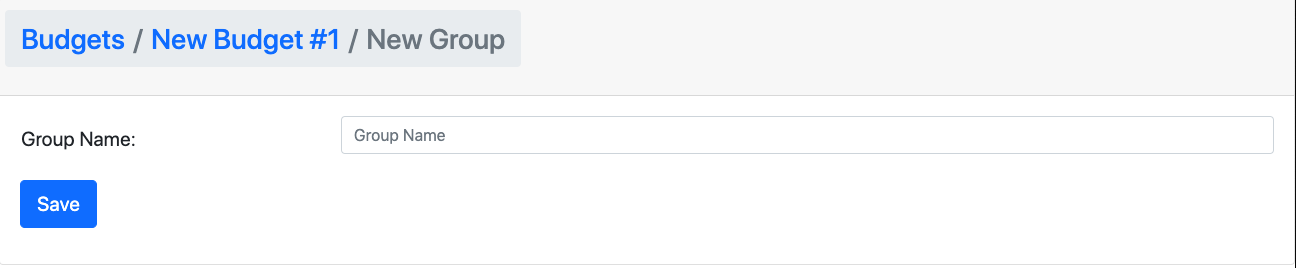
One budget has already been created already but to add another you would simply click the plus icon in the top right corner.

**Step 3: Create and Configure a group**



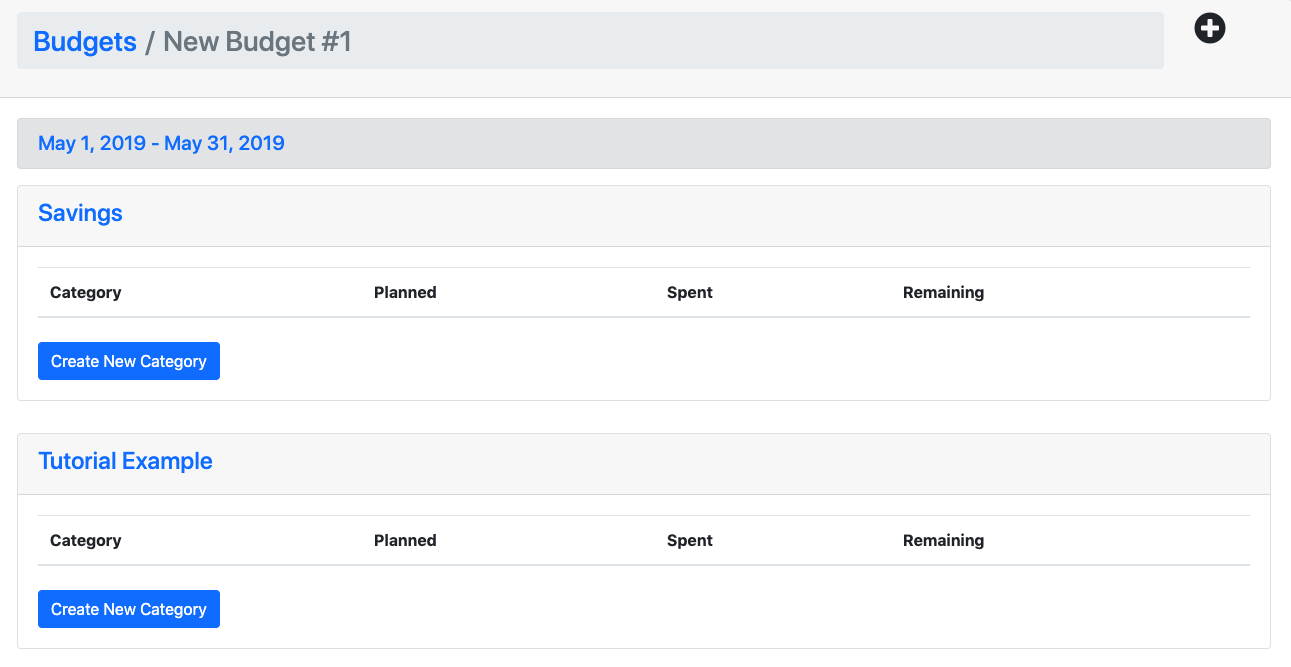
After clicking a budget, it will direct you to the budget endpoint. As you can see the user has a variety of options. Note the application tells you which budget you are in, the time period of the budget and automatically creates a Savings group already for them.

To create a new group, simply click the Create Group button.

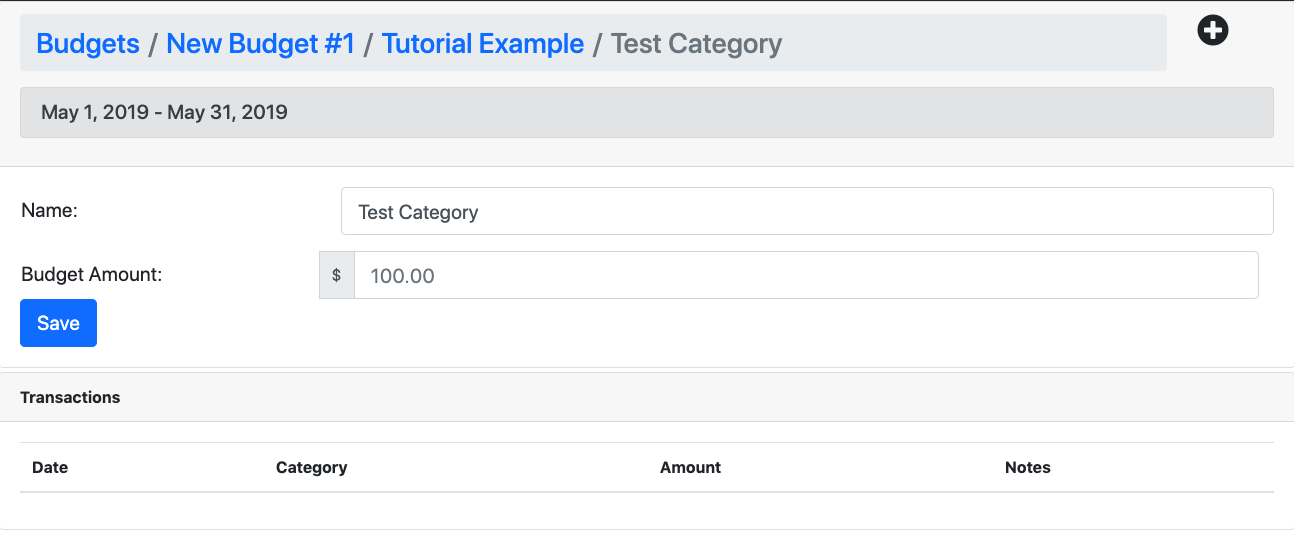


After doing so the application takes you to this page which only allows you to set the new group name. Enter a name and click save.

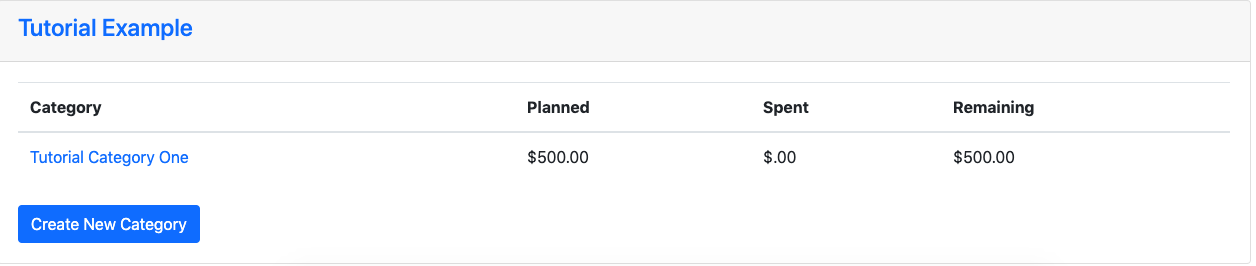
**Step 4: Create and add a category in your new group**



As you can see we now have a new group. Next we will add a category by clicking create new category in our new group.

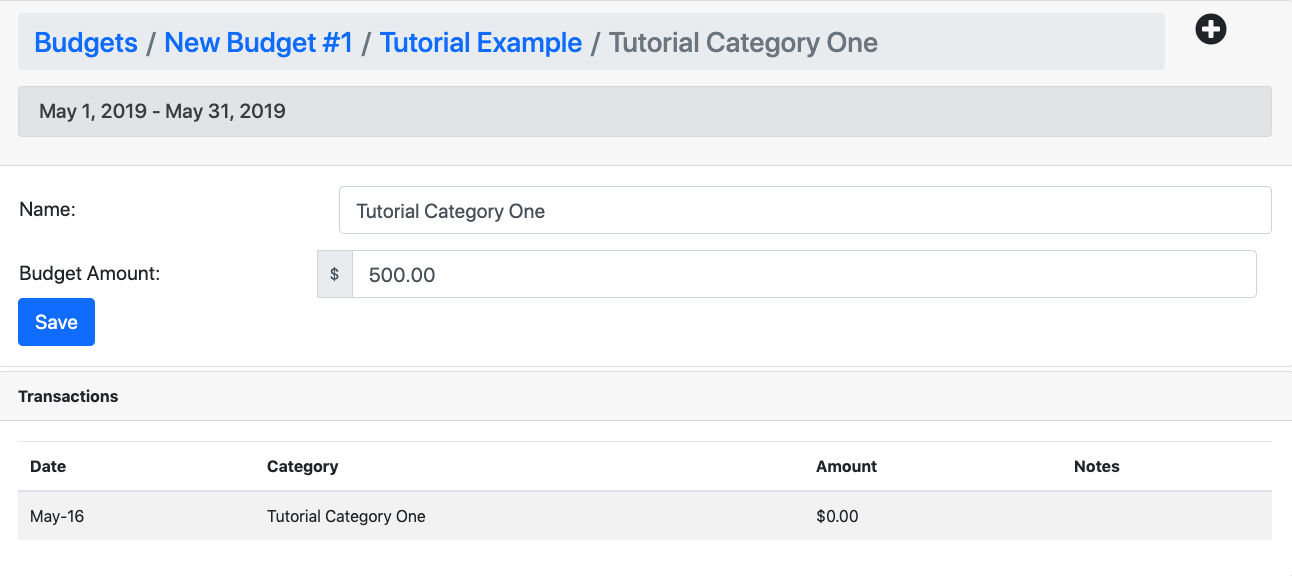


Once you have done that the application will allow you to set a name and a budget amount for the category. In this tutorial we will call it “Tutorial Category One” and give it a budget of $500.00.

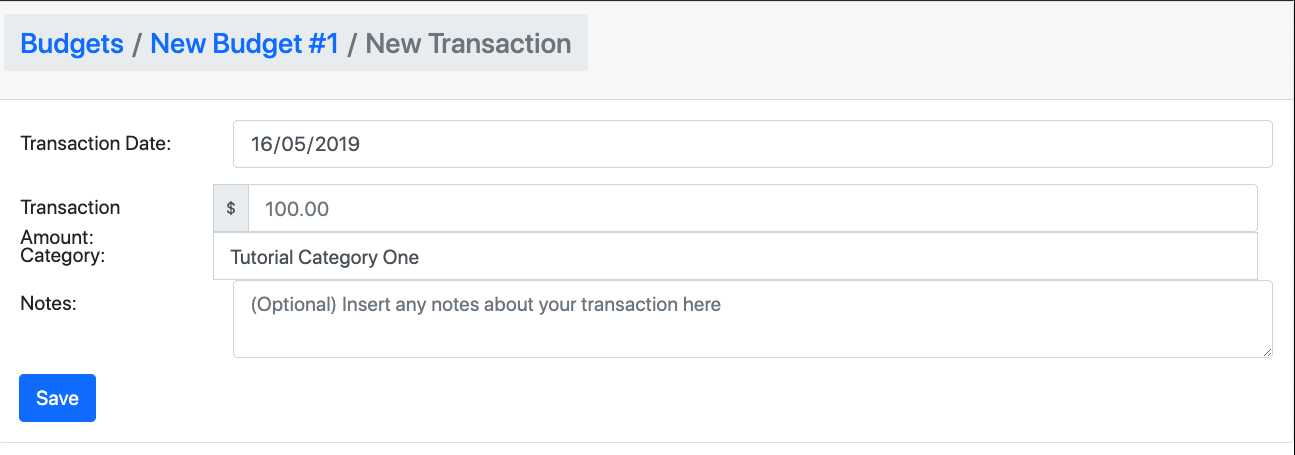


Now we have a category in our group.

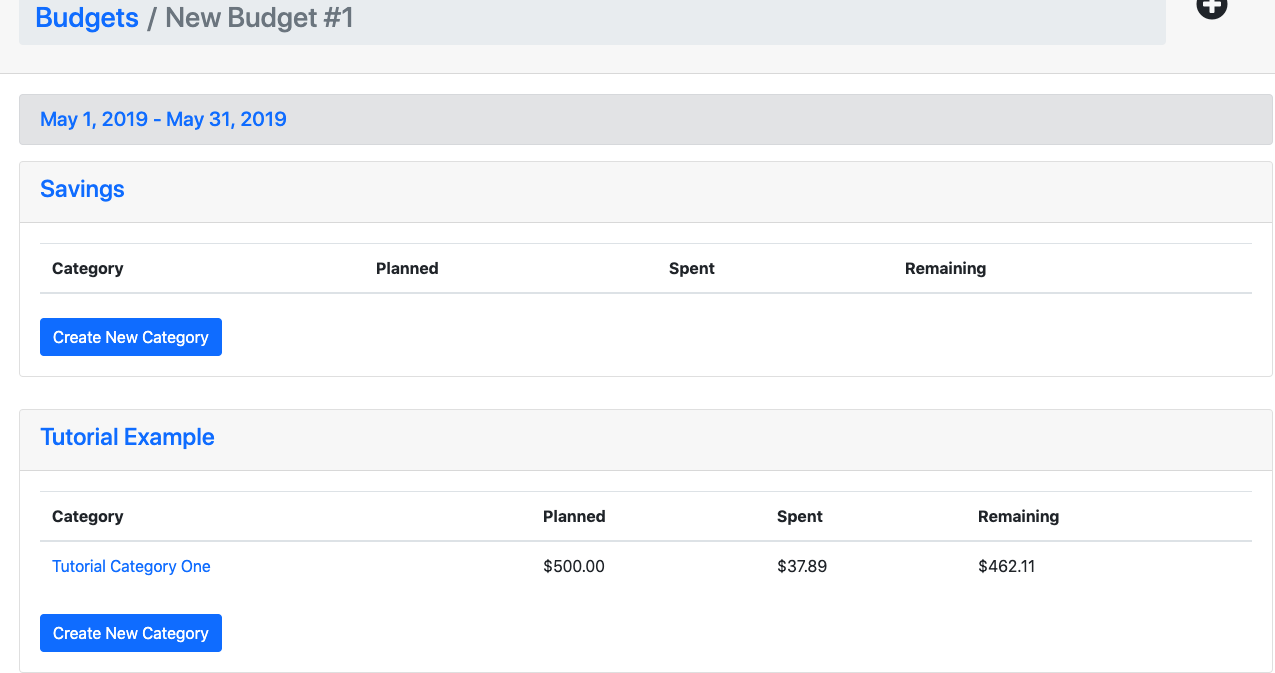
**Step 5: Add a transaction**

There are two ways to do this. The first method you must first click on the category you wish to add a transaction too. In our case we will click “Tutorial Category One”.

Recognize this screen? It is almost identical to the screen after we added a category. However, it is now its own and has our name and budget already in place. Add a transaction by clicking the plus icon in the top right corner.



Once you have clicked the plus icon it will direct you to this page. Here you can add a transaction by setting a date of transaction, amount, assign the category (which is already assigned because we clicked the category first and then any notes you may wish to add).



Once you save it the application will redirect you to the budget page with your updated information. It has automatically added your transaction to the spent column and taken that away from the budget you set previously.

Future Development

1. Future Development

Our aim is to develop the front end of the application so that the users can personalize budgets, groups and categories. A colour system would be very useful in terms of knowing exactly which budget you are under. The more personal it becomes they more comfortable the user will be using the application.

Another aspect we wish to develop further is the introduction of automatic incomes. The user should be able to create an income and then proportionally assign that income to each of the budgets meaning they will be in even more control of their finance’s. A long term goal would also be to implement a stocks feature that could potentially utilize an API to collect data regarding a user’s current stocks and values.

There is no current feature to delete a group or category without having direct access to the database. In the future the user should be allowed to remove a group if the desire. The database table is actually already set up to remove and associated transactions and categories but the front end does not currently cater for that yet.

Finally, they application would have to be completely bug free meaning, every link it mapped correctly, also if this were to be an application on the market for security reasons we would have to encrypt or user data in case of an illegal access to our database.