

**Design document**

**BudgetSimple**

|  |
| --- |
| **Date : 11.01.2022** |
| **Version : 1.1** |
| **Author : Stela Trencheva** |
| **Student number : 4520095** |

**Version history**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** | **State** |
| 1.0 | 08.10.2021 | Stela Trencheva | Initial version of the 4 level diagrams | Draft |
| 1.1 | 11.01.2022 | Stela Trencheva | Updated design document with more information and design decisions | Fianl release |

Contents

[1. Introduction 4](#_Toc92882987)

[2. Level 1: System Context diagram 5](#_Toc92882988)

[3. Level 2: Container diagram 7](#_Toc92882989)

[4. Level 3: Component diagram 9](#_Toc92882990)

[5. Level 4: Code 11](#_Toc92882991)

# Introduction

BudgetSimple is a full-stack web application for managing budget and money spend. There is a registration and login page where users can open an account with their information and start using the features of the application. Every user has an profile page to see/updated their information. Users can create wallets for different topics and with set budget. Every wallet owner can invite other members to join using a link or a QR code that is auto generated and unique for the wallet. Using the invitation link, other users of the application, can make a request to join and have to be accepted by a user that is already member of the wallet. When a new user is accepted, all other members are notified. In every wallet, users can change the budget, add transactions with description, category and amount, that can be divided between members equally or with individually specified amount, and see statistics of spending in wallet per member and per category with nice graphs. Additional thing that a user can do in BudgetSimple is rate the app and share their experience so that the admins of the application can see what users enjoy and what have to be improved. Administrators of the application can add different surveys about different topics they want to collect feedback on.

# Level 1: System Context diagram

The first level of the C4 diagram is an overview of the big picture of the software application and its users. It is usually shown to non-technical people.

Two different types of users have access to the BudgetSimple application – users and admins. All of them have different roles and access to the application features.

* Users – can create accounts and login in the application, manage their info, create wallets, invite people to their wallets and join wallets of other people, can add transactions to wallets, change wallet budget, track wallet spending per member and per category. They can also rate the application and complete surveys.
* Admins – can do all the actions that Users can, except rating the application, but they can also create surveys and see the ratings from the users

Intended audience are both technical and non-technical people, inside and outside of the software development team that have interest in tracking their spend.

Login in the application, manage their info, create wallets, invite people to their wallets and join wallets of other people, can add transactions to wallets, change wallet budget, track wallet spending per member and per category. They can also create surveys and see users’ raitngs.

Create accounts and login in the application, manage their info, create wallets, invite people to their wallets and join wallets of other people, can add transactions to wallets, change wallet budget, track wallet spending per member and per category. They can also rate the application and complete surveys.

**BudgetSimple User**[Person]  
A user of the system with personal account

**BudgetSimple Admin**[Person]  
A user of the system with personal account

**BudgetSimple**

[software system]

Allows users to track their finances and how they spent it.

**E-mail system**

[software system]

Allows the customer to receive

# Level 2: Container diagram

The second level of the C4 diagram shows the shape of the software architecture and the major technology choices. Also it describes how the containers communicate with one another. It is useful for software developers.

Backend and front end connection:

* REACT front end - it provides the UI and all of the website functionality within a browser environment. Users, admins and customer support employees use the interface to interact with the back-end of the application
* Back end - it provides all of the data and logical functionality of the website using Spring boot and Java. It makes the front-end functionality work
* Database - it is stores all content information and directly interacts with the back-end when a request for specific information is created (retrieving/adding/updating/deleting information).

The backend of the application is connected with the front-end using HTTP requests to connect to REST endpoints, allowing the performance of CRUD operations. The backend of the application is interacting with the database through JPA implementation (Hibernate). JPA/Hibernate was chosen after a short research on Java/Spring Boot database interaction methods. The results from the research are that JPA with Hibernate is the most popular and suitable for beginners as there are a lot of publicly available resources and tutorials.

The user interface of the application is fully developed using React, JavaScript with HTML and CSS. It consists of custom built components that represent different functionalities. React was chosen for the project, not only because it is required from the course, but also because it has fast learning curve and is widely used so there are many available resources. It also provides a component based structure which allows to easily build a larger application using reusable components.

Spring Boot and Java are the technologies used for the implementation of the backend. Spring Boot’s main purpose in the application is to reduce overall development time and increase efficiency by having a default setup for unit and integration tests and it also let beginners start with a Java application easier and quicker.

**BudgetSimple Customer support**[Person]

**BudgetSimple Admin**[Person]

**BudgetSimple SPA**[Container: JavaScript and React]  
Provides all the BudgetSimple functionalities from the Web Application via their web browser.

Functionalities shown for the role of: **User, Customer support, Admin**

Sends an email with auto generated password every time a new customer support is created

Sends an email with auto generated password when a user has forgotten their password

**BudgetSimple User**[Person]

Makes API calls to  
[JSON/HTTPS]

**API Application**[Container: Java and Spring Boot]

Provides users, wallets, bonuses, chat functionalities via a JSON/HTTPS API.

Sends email to

**E-mail system**

[software system]

Allows the customer to receive emails

Visits budgetsimple.com   
(HTTPS)

Reads from and writes to

Stores users, wallets, transactions, bonuses, ratings and messages.

**Database**[Container: Docker, MySQL database]

# Level 3: Component diagram

Components in the BudgetSimple application are:

* Controllers – used to create endpoints of the application which can be used to connect to the frontend. The frontend of the application uses the endpoints in the controllers to get the needed information from the backend.
* Service classes – All the logic in the application is implemented in the service classes. There the information from the database is manipulated and ready to be used from the controllers.
* Data access layer – In this part of the application is implemented the connection with the database. Information is retrieved from or written to the database

The service components are connected with the database components using interfaces which allows flexibility in the change of the data type. All of the classes use Data Transfer Objects (DTO).

Makes API calls to   
[JSON/HTTPS]

**BudgetSimple SPA**[Container: JavaScript and React]  
Provides all the BudgetSimple functionalities from the Web Application via their web browser.

**Controllers**(Container: Spring Boot REST Controller)

**DTOs**

Uses

**Database**[Container: Docker, MySQL database]

**Data Access Layer**

**<<Interface>>**

**IUserData**

**Services**

Stores products, customers and orders information.

# Level 4: Code

\*Link to the UML diagram

****