

**Курсов проект**

Prosplitter

**Дисциплина: Fullstack Application Development with Node.js + Express.js + React.js**

**Версия 1.0**

|  |  |
| --- | --- |
| Фак. № | Име на студент |
| 25737 | **Мартин Христов** |
| 25735 | **Станислав Стойнов** |

**Юни, 2018**

1. **Съдържание**

[1](#_gjdgxs) ВЪВЕДЕНИЕ 4

[1.1](#_30j0zll) Цел 4

[1.2](#_1fob9te) Обхват 4

[1.3](#_3znysh7) Резюме 4

[1.4](#_2et92p0) Дефиниции и акроними 5

[2](#_tyjcwt) АНАЛИЗ НА ИЗИСКВАНИЯТА 6

[2.1](#_3dy6vkm) Функционални изисквания 6

[2.2](#_1t3h5sf) Нефункционални изисквания 7

[2.2.1](#_4d34og8) Изисквания към потребителския интерфейс 7

[2.2.2](#_2s8eyo1) Производителност 7

[2.2.3](#_17dp8vu) Наличност 7

[2.2.4](#_3rdcrjn) Сигурност 7

[2.2.5](#_26in1rg) Съответствие със стандарти 7

[3](#_35nkun2) СИСТЕМНА АРХИТЕКТУРА 8

[3.1](#_1ksv4uv) Архитектурно проектиране 8

[3.1.1](#_44sinio) Front-end - JavaScript + HTML + CSS 8

[3.1.2](#_2jxsxqh) NodeJS (Client side) 8

[3.1.3](#_z337ya) Ruby (Server side) 8

[3.1.4](#_3j2qqm3) База от данни 8

[3.2](#_1y810tw) Описание на декомпозицията 9

[3.2.1](#_4i7ojhp) Front-end част 9

[3.2.2](#_2xcytpi) Server side (NodeJS) 9

[3.2.3](#_1ci93xb) Server side ([Ruby](#_z337ya)[) 9](#_1ci93xb)

[3.3](#_3whwml4) Обосновка на проектирането 10

[4](#_2bn6wsx) ДАННОВО ПРОЕКТИРАНЕ 12

[4.1](#_qsh70q) Описание на данните 12

[4.2](#_3as4poj) Речник на данните 12

[4.2.1](#_1pxezwc) Потребител (User): 12

[4.2.2](#_49x2ik5) Типове потребители (User Type) 12

[4.2.3](#_2p2csry) Проекти (Project): 13

[4.2.4](#_147n2zr) Състояние на проекти (Project Status): 13

[4.2.5](#_3o7alnk) Маркери на проекти (Project tag): 13

[4.2.6](#_23ckvvd) Вписани потребители към проекти (Project subscriber) 13

[4.2.7](#_ihv636) Канали към проекти (Project channel) 13

[4.2.8](#_32hioqz) Съобщения (Messages) 14

[4.2.9](#_1hmsyys) Журнали (Log) 14

[4.2.10](#_41mghml) Действия (Action) 14

[5](#_2grqrue) КОМПОНЕНТНО ПРОЕКТИРАНЕ 15

[6](#_vx1227) ПРОЕКТИРАНЕ НА ПОТРЕБИТЕЛСКИЯ ИНТЕРФЕЙС 16

[6.1](#_3fwokq0) Обобщение на потребителския интерфейс 16

[6.2](#_4f1mdlm) Екранни изображения 16

# ВЪВЕДЕНИЕ

## Цел

Целта на документа е да се представи софтуерна система описваща взаимодействието между потребители и група потребители. Идеята е да се създаде платформа, която помага на потребителите да споделят разходите си с приятели. Чрез системата лесно ще се следят всички разходи направени по време на пътуване, домакински разходи или излизане на ресторант.

## Обхват

Софтуерната система ще обхване всички потребители без значение от възраст, пол, местоположение или расова принадлежност, които искат да са организирани в разходите си по време на някакви активност (пример: почивка, храна, кола и тн.).

## Резюме

В текущия документ е разгледана софтуерна система, служеща за следенето на разходи на отделни потребители. Чрез лесния и иновативен интерфейс на платформата, тя предоставя удобен начин за създаване на екип с ясни цели и отговорности. За реализацията на проекта са включени различни софтуерни архитектури и технологии, позволяващи бързо, сигурно и надеждно създаване на стабилно приложение. Чрез използване на разпределени системи в архитектурата на плаформата се гарантира равномерното натоварване на системата. В проекта са включени следните компоненти:

* Потребителски интерфейс
* Клиентсака част
* Сървърна част
* Комуникация (потребител - клиентска част, клиентска част - сървърна част)
* База данни

Дефиниции и акроними

* Client-site (**CSL**) - Слой на система, при който потребителя комуникира директно. Той се грижи за визуализация на предоставените данни. Други възможни операции са: филтриране, нормализиране и хидриране на данните;
* Server-site (**SSL**) - Слой, който получава получените данни, прилага някаква бизнес логика. След приложените процеси, той съхранява данните в база данни;

# АНАЛИЗ НА ИЗИСКВАНИЯТА

## Функционални изисквания

Примерен формат на таблица за описание на функционални изисквания.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Изискване | Коментар | Приоритет |
| FR\_01 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Нефункционални изисквания

### Изисквания към потребителския интерфейс

Интуитивния, прост за използване интерфейс би накарал потребителя да използва отново тази система. Един потребител трябва да може да свикне с интерфейса на системата за по-малко от 5 минути.

Потребителският интерфейс на системата позволява бърз и лесен достъп до всяка една от секциите ѝ:

* публикуване на проект;
* разглеждане на проекти;
* профил;

Системата е подходяща за всякакви платформи и резолюции. Всички съобщения за грешки се прихващат от системата и се поднасят на потребителя чрез интерактивен и удобен потребителски интерфейс.

### Производителност

Такъв тип интерфейси се нуждаят от висока скорост на интеракция. Всички потребителски дейстия трябва да бъдат обработвани за под 300 ms.

### Наличност

Надеждността е основно качество при такива системи. Търси се успешно приемане на заявки в 99.99% от случаите.

Данните от системата ще се репликират на две различни места, за да се предотврати изгубването на информация.

### Сигурност

Надеждността е основно качество при такива системи. Търси се успешно приемане на заявки в 99.99% от случаите. Данните от системата ще се репликират на две различни места, за да се предотврати изгубването на информация.

### Съответствие със стандарти

Софтуерните системи са криптографски защитени. Достъпът до тях става, чрез SSH генериран ключ по алгоритъма (RSA **Rivest–Shamir–Adleman**). Размерът на ключа е 4096 бита. Всеки един от администраторите на системите ще разполага с по един ключ. Всеки ключ ще има допълнителна защита с парола.

#### Интернационализация (Internationalization) - W3C

Системата се поддържа на 2 езика - български и английски.

#### Достъпност (Accessibility) - W3C

Системата е напълно достъпна и позволява работене с нея от хора с увредено зрение, като поддържа различни видове Screen Reader инструменти. Например JAWS.

# СИСТЕМНА АРХИТЕКТУРА

## Архитектурно проектиране

Платформата се гради на няколко основни части:

### Front-end - JavaScript + HTML + CSS + React / Redux

Тук се съдържа цялата логика за потребителския интерфейс, интернационализация, глобализация и достъпност.

### NodeJS (Client side)

NodeJS backed сървиса сервира през REST информацията, която трябва да се покаже в системата.

### Nodejs + Express (Server side)

Чрез бизнес правила, дефинифирани в Server side системата, тя запазва информация от и към базата от данни.

### База от данни

Нерелационна база от данни, която пази всичката информация за различните потребители, лист от приятели, групи, сметки или активност. Например - имена и хеширани пароли и т.н.

## Описание на декомпозицията

### Front-end част

Описва как изглежда приложението. Ползва изцяло JavaScript, HTML и CSS.

### Server side (NodeJS)

Сервира следните REST endpoints:

### Server side (Nodejs)

#### Бизнес логика

Този слой обработва постъпилите данни по RPC-протокола. Прилага различни бизнес правила и записва данните.

Възможно е да постъпи заявка, която да изисква извличане на данни от слой “Съхранение”. При този случай се създава заявка към слоя управляващ бази данни и се събират необходимите ресурси поискани от Client-Side.

#### Слой Съхранение (Storage Layer)

Service-side система кореспондира с база данни (DB) по следния начин:

#### Слой “Комуникация”

Данните се предават/получават посредством REST от/към (Server-side / Client-Side).

#### Слой “Аутентификация”

При познат потребител с определени права за достъп до системата се генерира token. Той се изпраща на Client-side. При всяка заявка се проверява - дали подадения token е изтекъл **[RFC6749]**.

# ПРОЕКТИРАНЕ НА ПОТРЕБИТЕЛСКИЯ ИНТЕРФЕЙС

## Обобщение на потребителския интерфейс

След постъпването на адреса на Prosplitter в интернет.

## Екранни изображения