Università degli Studi di Palermo
Corso di laurea in Ingegneria Informativa a.a.2020/2021
Ingegneria e Progettazione del Software
Programmazione Web e Mobile

# OBJECT DESIGN DOCUMENT


Gruppo GAAS
Marianna Francesca Amalfi
Vittorio Sanfilippo
Giuseppe Gullo
Michele Abanese

# ODD

# (Object Design Document)

## Indice

# 1. Compromessi e scelte nella progettazione degli Oggetti

## 1.1. Suddivisione in Livelli

Al fine di garantire la modularità del sistema e per facilitarne lo sviluppo, il software è stato diviso in livelli differenti. Lo sviluppo è semplificato in quanto le componenti del sistema, in questo modo, vengono sviluppate autonomamente rispetto alle altre. Inoltre, ciò facilita l'identificazione di problemi del sistema in fase di implementazione.
I livelli da noi identificati sono:

- Gestione delle funzionalità di base del sistema mediante API, React ed Express (Model)
- Interfaccia grafica, gestita mediante React e Javascript (View)
- Logica di Sistema gestita mediante Node.js ed Express (Controller)
- DBMS

## 1.2. Gestione e progettazione dell'interfaccia grafica

Per la gestione delle interfacce grafiche si è scelto di utilizzare un'architettura web-based mediante **JavaScript** e il framework di sviluppo **ReactJS**, una libreria Javascript sviluppata da Facebook per la creazione di User Interface interattive, in quanto facilita la realizzazione di interfacce dinamiche in funzione dei dati utilizzati.
Inoltre si è scelto di affidarsi al framework **CSS Bootstrap** per quanto riguardo la realizzazione dei componenti grafici, in quanto consentono la realizzazione di interfacce responsive in modo rapido (pur non trascurando la loro resa grafica), al fine di rendere disponibile il sistema su piattaforme differenti (rendendolo multipiattaforma).
Inoltre per la realizzazione di alcuni componenti grafici si è utilizzata la libreria **JQuery** per applicazioni web la quale semplifica l'utilizzo di JavaScript facilitando la manipolazione di elementi DOM in pagine HTML.
Per la realizzazione di richieste HTTP è stato scelto **axios**, un client HTTP promise-based che facilita l'implementazione di richieste asincrone.
Per impostare il file di routing al fine di reindirizzare i componenti è stato utilizzato il modulo **react-router-dom.**
Per l'inserimento di icone all'interno dell'applicativo web e mobile si è fatto riferimento alle API di Google quali **Icons** di **Google Fonts** le quali hanno permesso il loro inserimento mediante semplici snippets HTML.

I motivi e i vantaggi di queste scelte sono i seguenti:

- Incremento della modularità del sistema mediante React, il quale permette la costruzione di interfacce complesse mediante l'aggregazione di "componenti" differenti.
- Riuso facilitato, grazie alla suddivisione in componenti.
- Gestione degli eventi semplificata, mediante le funzionalità di Javascript.
- Progettazione agevole del layout, mediante **JSX** (la cui sintassi è simile, per certi versi, a quella del linguaggio di markup HTML. JSX, inoltra, sfrutta anche le potenzialità di Javascript), CSS, JavaScript e i componenti offerti da Bootstrap.
- Indipendenza dal sistema con cui l'utente accede al software (il sistema è, di fatto, multipiattaforma).

Le boundary del sistema presenti nel RAD sono state implementate mediante pagine JSX, le quali comunicano e interagiscono in modo dinamico con il sistema mediante i rispettivi controller e mediante le API di Javascript. Le Control precedentemente individuate sono state mappate nelle rotte di Express. Invece per le Entity si è scelto di utilizzate gli Hooks di React. Questi permettono di utilizzare lo stato ed altre funzionalità di React senza dover scrivere classi.
Di seguito sono riportati tutti i pacchetti usati per lo sviluppo e la gestione del Front-End:

- "@babel/core": "^7.12.10"
- "@babel/node": "^7.12.10"
- "@babel/preset-env": "^7.12.11"
- "@babel/preset-react": "^7.12.10"
- "@date-io/date-fns": "^1.3.13"
- "@material-ui/core": "^4.11.4"
- "@material-ui/icons": "^4.11.2"
- "@material-ui/pickers": "^3.3.10"
- "axios": "^0.21.1"
- "babel-loader": "^8.2.2"
- "clean-webpack-plugin": "^3.0.0"
- "date-fns": "^2.22.1"
- "dotenv-webpack": "^6.0.0"
- "emailjs-com": "^3.1.0"
- "express": "^4.17.1"
- "fs-extra": "^9.1.0"
- "html-webpack-harddisk-plugin": "^2.0.0"
- "html-webpack-plugin": "^5.3.1"
- "immer": "^8.0.1"
- "npm-run-all": "^4.1.5"

- "react": "^17.0.1"
- "react-dom": "^17.0.1"
- "react-flatpickr": "^3.10.7"
- "react-markdown": "^5.0.3"
- "react-router-dom": "^5.2.0"
- "react-tooltip": "^4.2.13"
- "react-transition-group": "^4.4.1"
- "socket.io-client": "^3.1.0"
- "use-immer": "^0.4.2"
- "webpack": "^5.24.4"
- "webpack-cli": "^4.5.0"
- "webpack-dev-server": "^3.11.2"

## 1.3. Gestione e progettazione della componente server

Per la gestione della componente server si è scelto di utilizzare **Node.js**, un ambiente a runtime Javascript comandato da eventi asincroni, che risulta particolarmente appropriato per lo sviluppo di applicazioni di rete scalabili, in quanto è orientato agli eventi: pertanto, le operazioni di I/O possono essere svolte in modo asincrono. Grazie a questo, risulta un ambiente altamente scalabile e che garantisce un alto throughput.
Nello specifico, come framework per lo sviluppo web è stato scelto **Express.js**.
Esso facilita la gestione delle richieste HTTP, in quanto permette di configurare i middleware per interagire con le richieste HTTP (effettuate tramite axios) del front-end mediante diversi URL (routes). Express permette di rispondere in modo dinamico alle pagine JSX e ai component JS.
Di seguito sono riportati tutti i pacchetti usati per lo sviluppo e la gestione del Back-End:

- "@marketto/codice-fiscale-utils": "^2.0.3"
- "bcryptjs": "^2.4.3"
- "cors": "^2.8.5"
- "dotenv": "^8.2.0"
- "express": "^4.17.1"
- "express-fileupload": "^1.2.1"
- "is-valid-birthdate": "^0.1.0"
- "jsonwebtoken": "^8.5.1"
- "md5": "^2.3.0"
- "mongodb": "^3.6.3"
- "multer": "^1.4.2"

- "nodemon": "^2.0.7"
- "password-validator": "^5.1.1"
- "path": "^0.12.7",
- "sanitize-html": "^2.3.0"
- "socket.io": "^3.1.0"
- "validator": "^13.5.2"

## 1.4. Gestione e progettazione della componente mobile

L'interfaccia mobile è realizzata tramite una **WebView**, la quale permette di visualizzare pagine web. La sua realizzazione è resa possibile dalla libreria **WebKit** inclusa in **Android**. Attraverso essa, è possibile quindi integrare una applicazione web all'interno di dispositivo mobile, permettendone un uso completo grazie alla natura **responsive** e **multipiattaforma** del software.

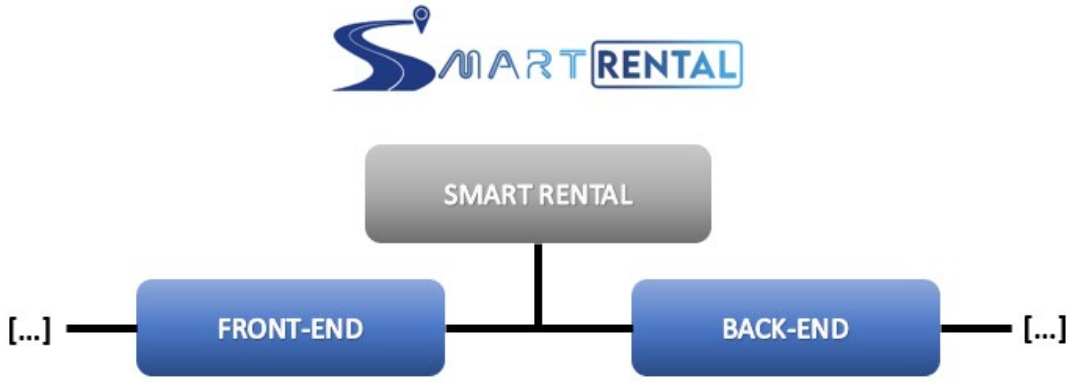
## 1.5. Connessione al Database

Per interfacciarsi al database si è creato un middleware, denominato come "**db**" che fornisce metodi per la gestione la connessione al DBMS che si è deciso implementare mediante MongoDB. Per effettuare le query, invece, è stato scelto di attenersi al paradigma di progettazione di Express che prevede l'utilizzo dei cosiddetti Model. Avendo scelto in fase implementativa il pattern MVC, i metodi costituenti la DBMSBoundary sono stati distribuiti nei vari model definiti.

# 2. Package

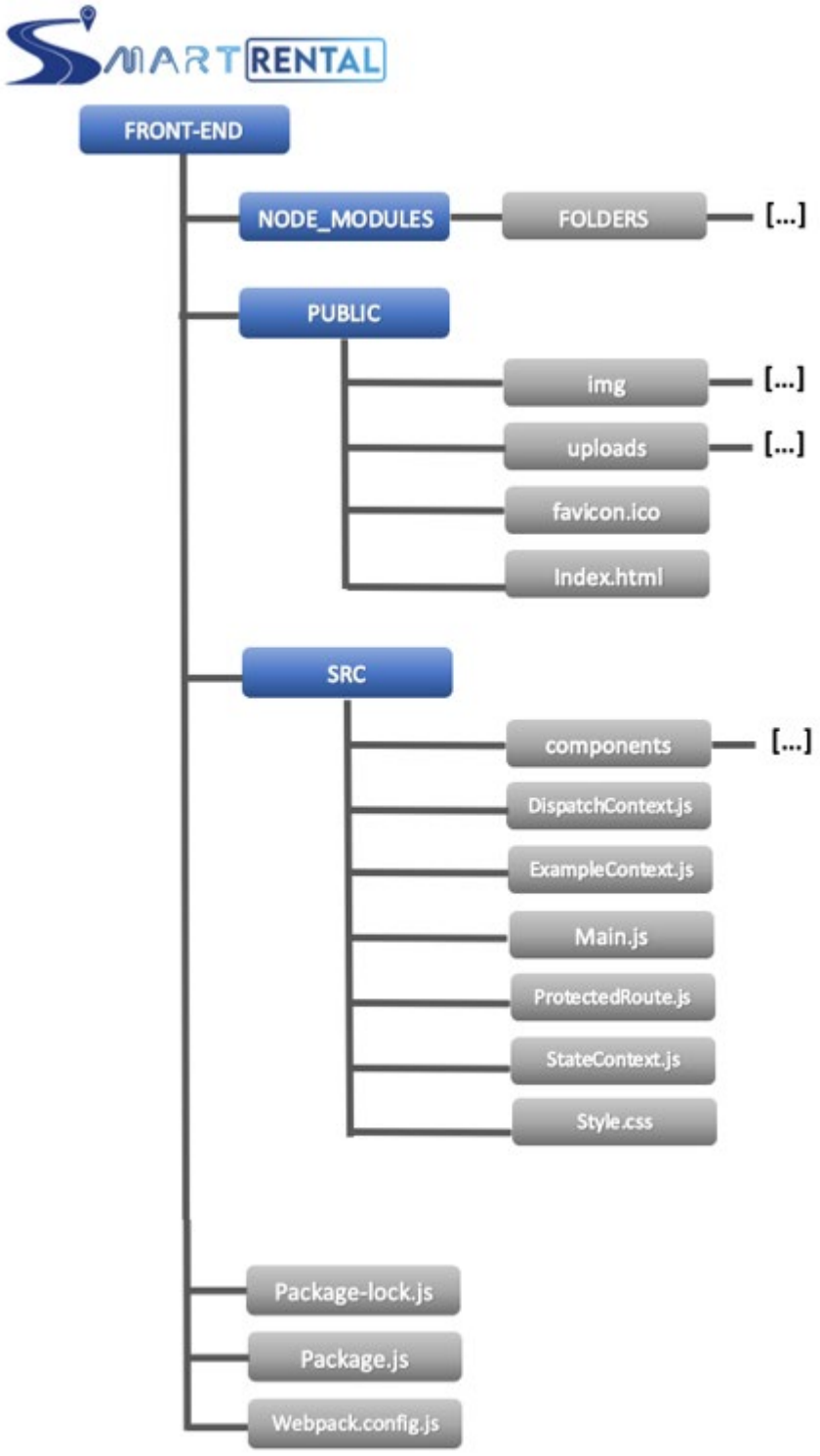
## 2.1. Directory

Descrizione del Package del Progetto SmartRental© il quale si suddivide in Front-End e Back-End



## 2.2. Front-End

Descrizione del Package del Progetto SmartRental©, lato Front-End.
Sono stati esclusi i singoli file per garantire una visualizzazione semplice a causa della loro mole.



## 2.3. Back-End

Descrizione del Package del Progetto SmartRental©, lato Front-End.
Sono stati esclusi i singoli file per garantire una visualizzazione semplice a causa della loro mole.