Package Management

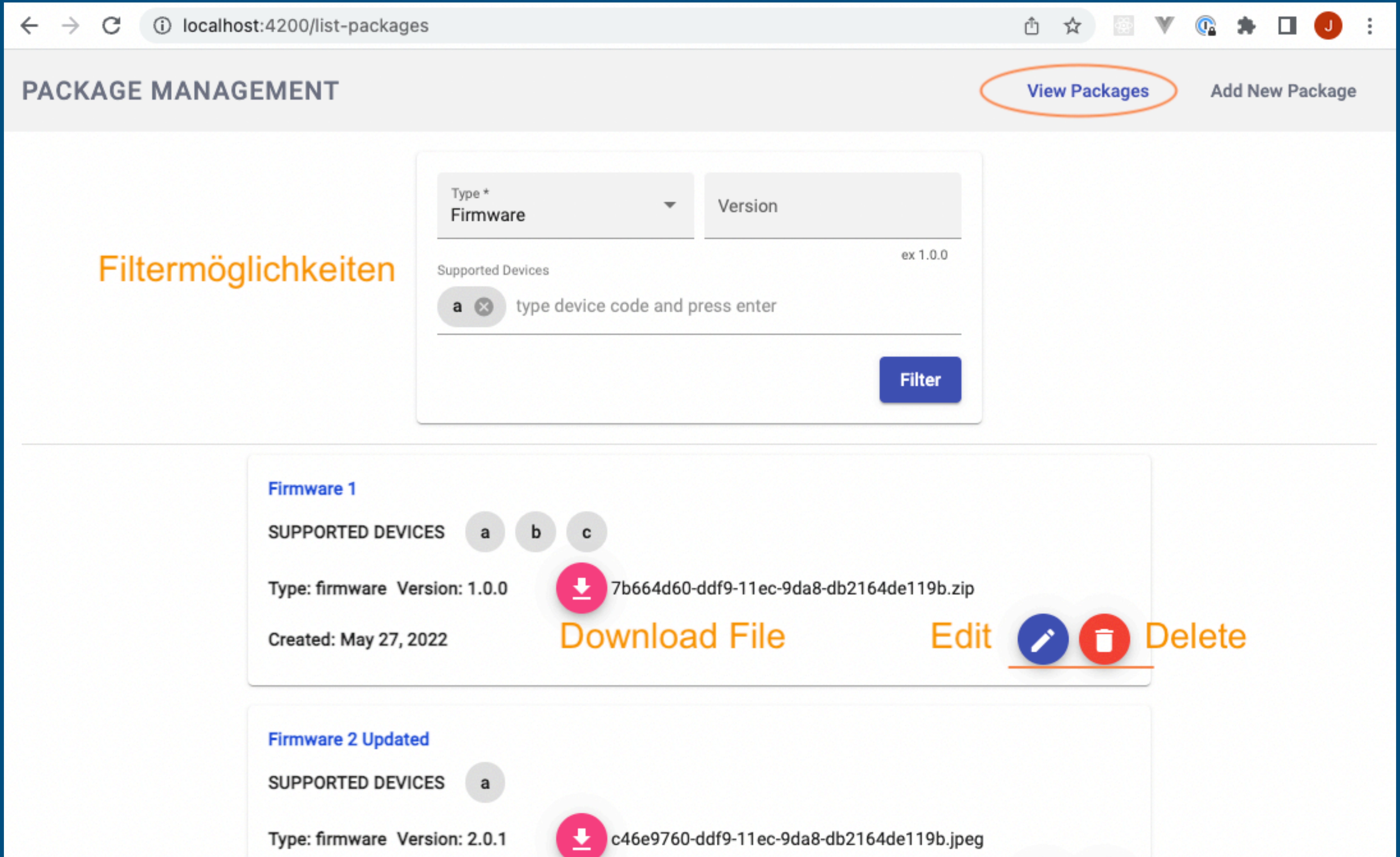
Demo-Projekt

João Aroeira

Beschreibung

- Das Projekt besteht aus einem Paketmanager, in dem es möglich ist, Pakete zu registrieren, die Metadaten und eine Datei enthalten.
- Ein Paket enthält die folgenden Metadaten: Title, Type (Firmware oder Tool), Version und Supported Devices.
- Alle CRUD-Operationen werden implementiert. Außerdem kann die Liste der Pakete nach ihren Metadaten gefiltert werden.

User Interface



localhost:4200/list-packages

PACKAGE MANAGEMENT

View Packages | Add New Package

Filtermöglichkeiten

Type *: Firmware

Version (ex 1.0.0)

Supported Devices: a — type device code and press enter

Filter

Firmware 1

SUPPORTED DEVICES a b c

Type: firmware Version: 1.0.0 — 7b664d60-ddf9-11ec-9da8-db2164de119b.zip

Created: May 27, 2022

Download File

Edit

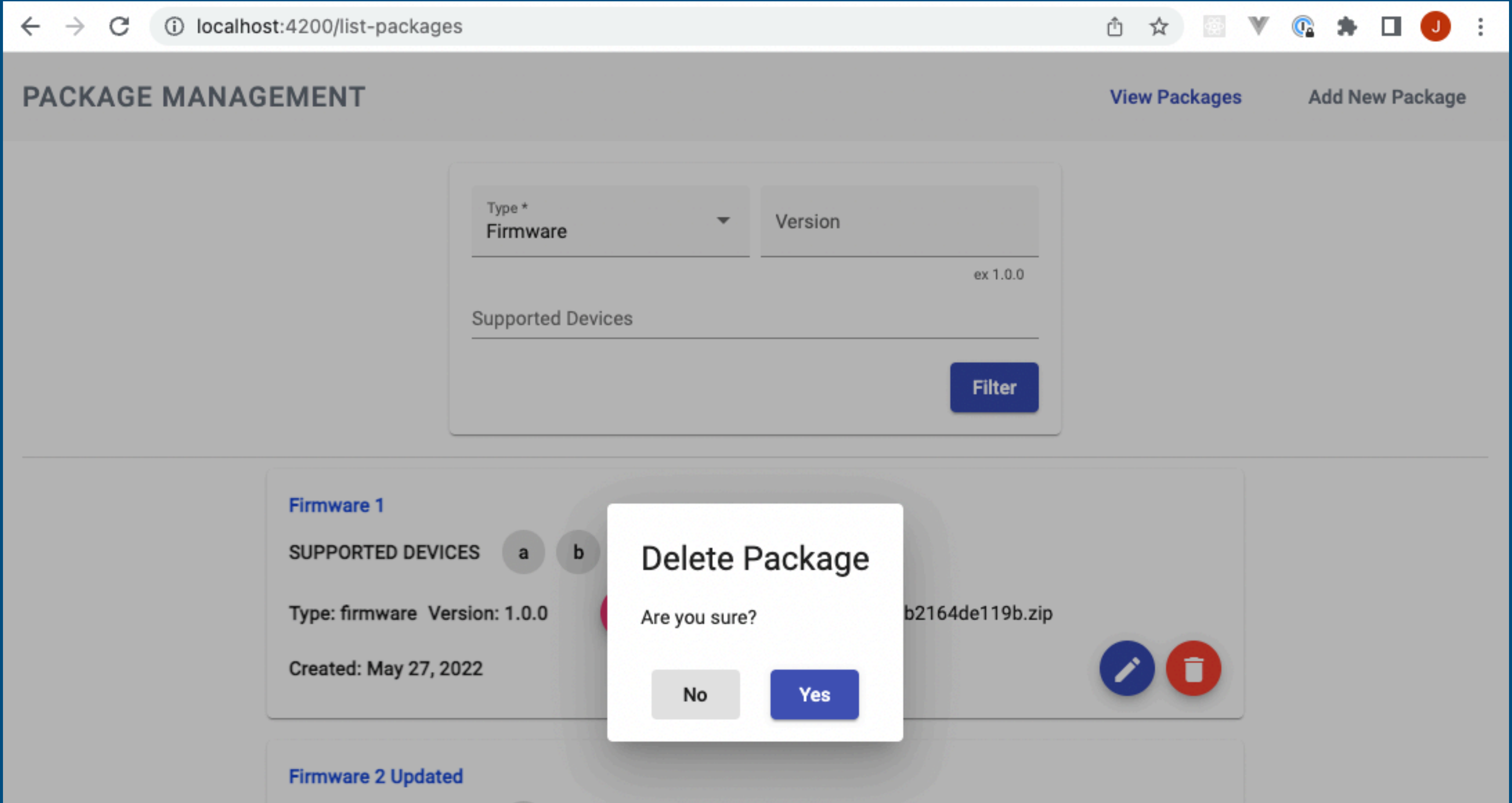
Delete

Firmware 2 Updated

SUPPORTED DEVICES a

Type: firmware Version: 2.0.1 — c46e9760-ddf9-11ec-9da8-db2164de119b.jpeg

User Interface



localhost:4200/list-packages

PACKAGE MANAGEMENT

View Packages

Add New Package

Type *

Firmware

Version

ex 1.0.0

Supported Devices

Filter

Firmware 1

SUPPORTED DEVICES a b

Type: firmware Version: 1.0.0

b2164de119b.zip

Created: May 27, 2022

Firmware 2 Updated

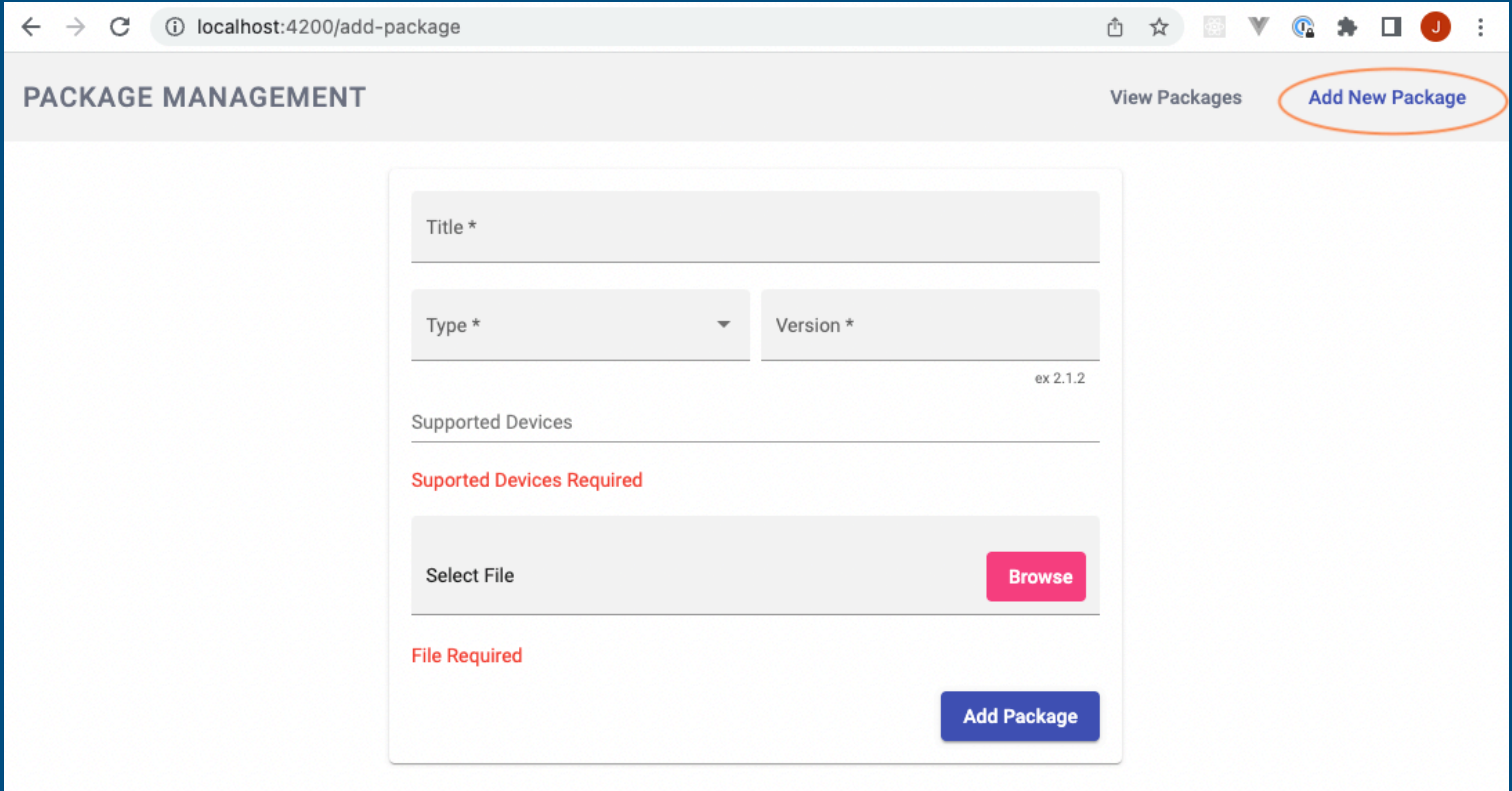
Delete Package

Are you sure?

No

Yes

User Interface



localhost:4200/add-package

PACKAGE MANAGEMENT

View Packages

Add New Package

Title *

Type *

Version *

ex 2.1.2

Supported Devices

Suported Devices Required

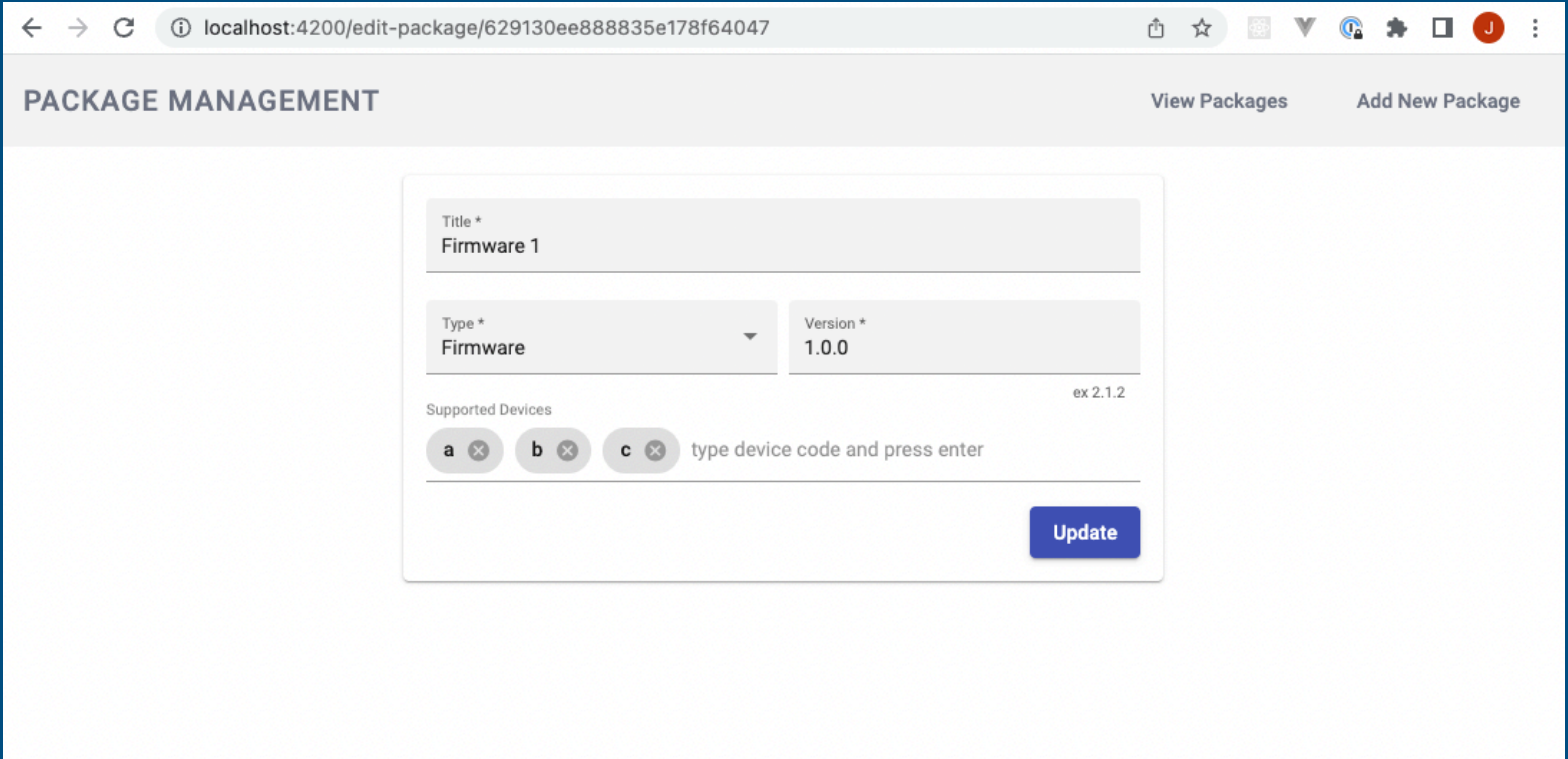
Select File

Browse

File Required

Add Package

User Interface



localhost:4200/edit-package/629130ee888835e178f64047

PACKAGE MANAGEMENT

View Packages

Add New Package

Title *
Firmware 1

Type *
Firmware

Version *
1.0.0

ex 2.1.2

Supported Devices

a b c type device code and press enter

Update

Technologie-Stack

MEAN STACK

mongoDB®

Express

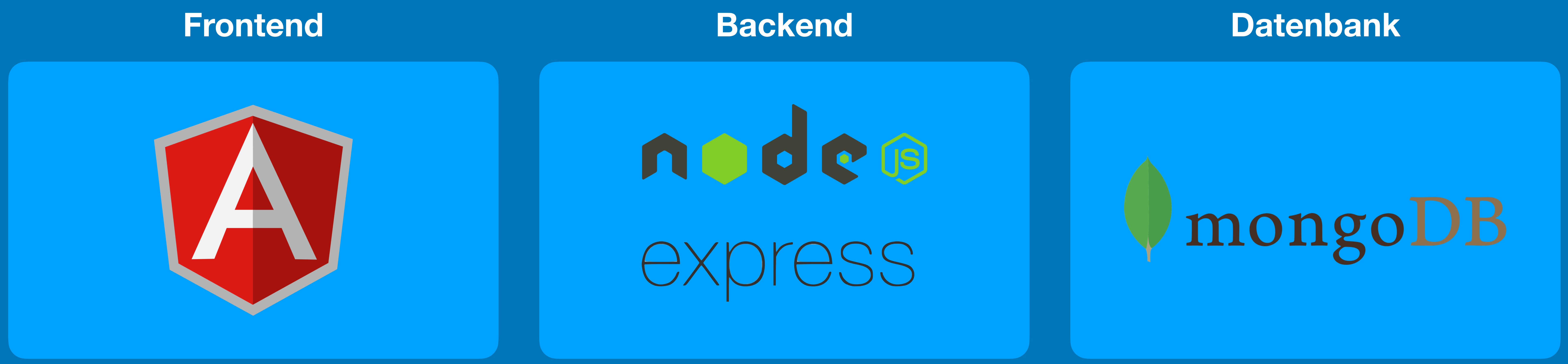
A

node

JS®

docker

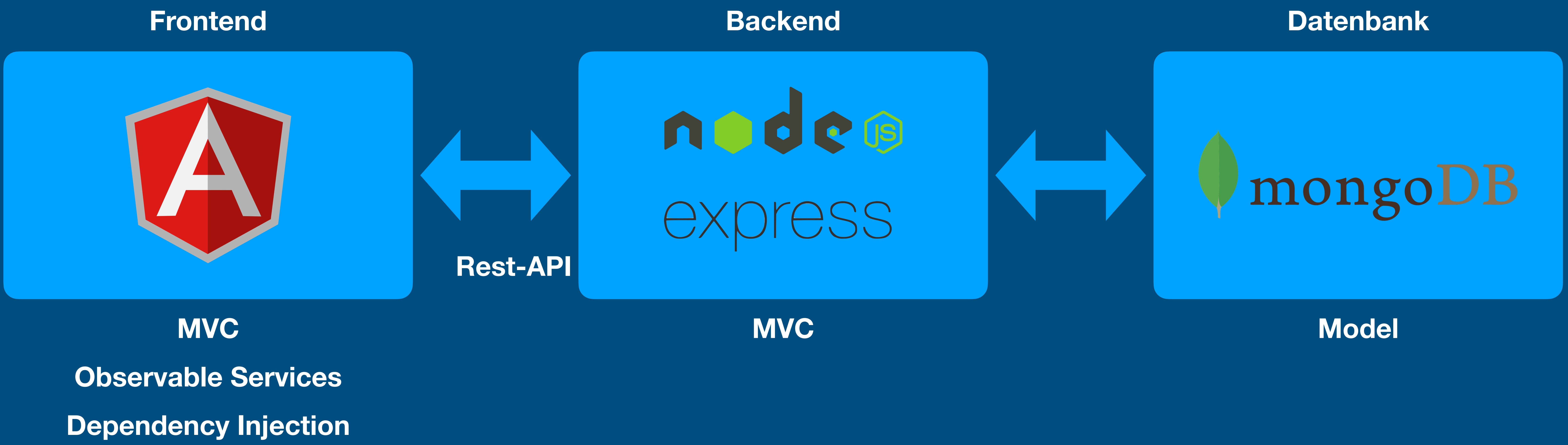
Docker-Compose

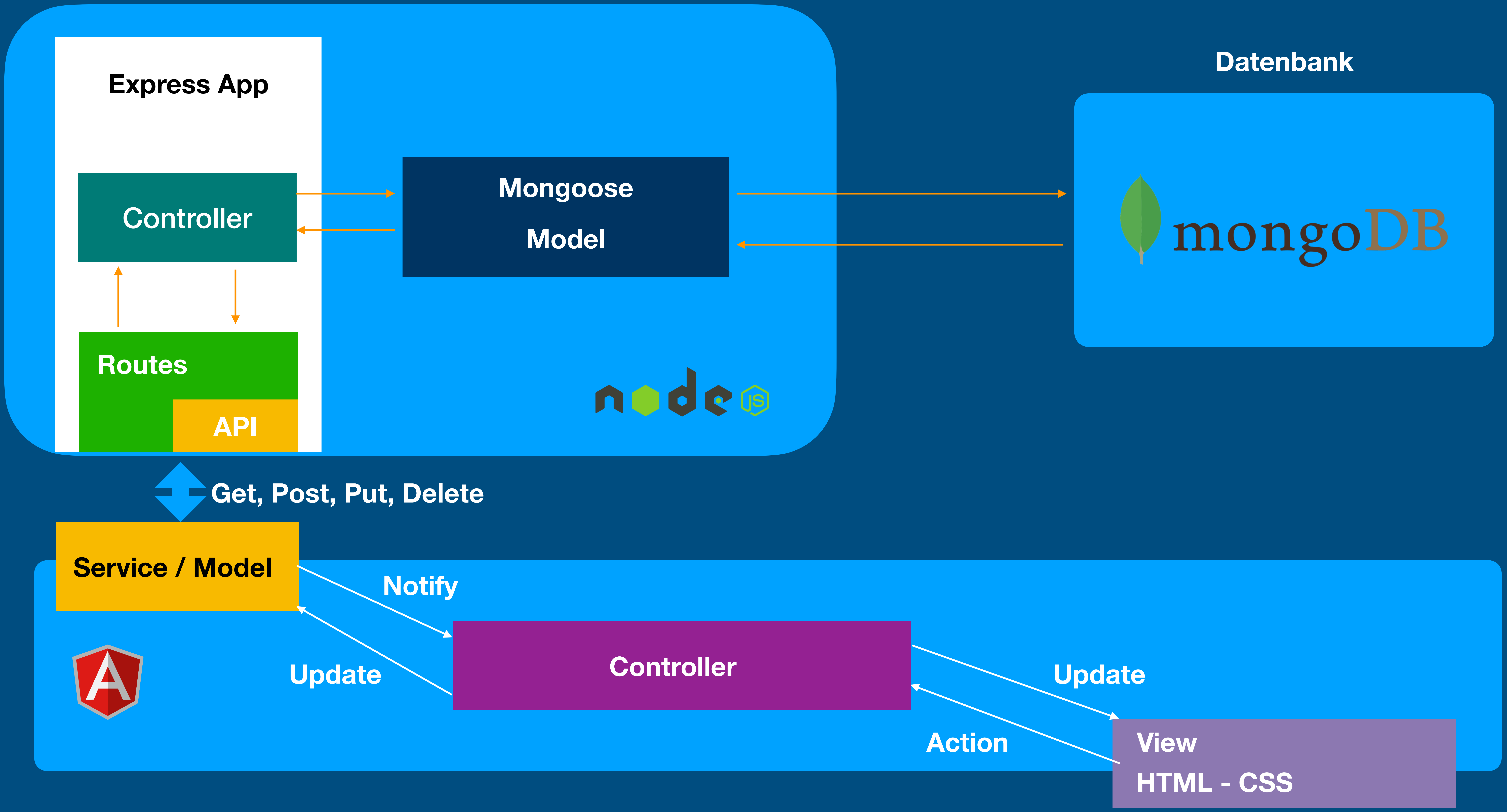


Software-Architektur

Design Patterns

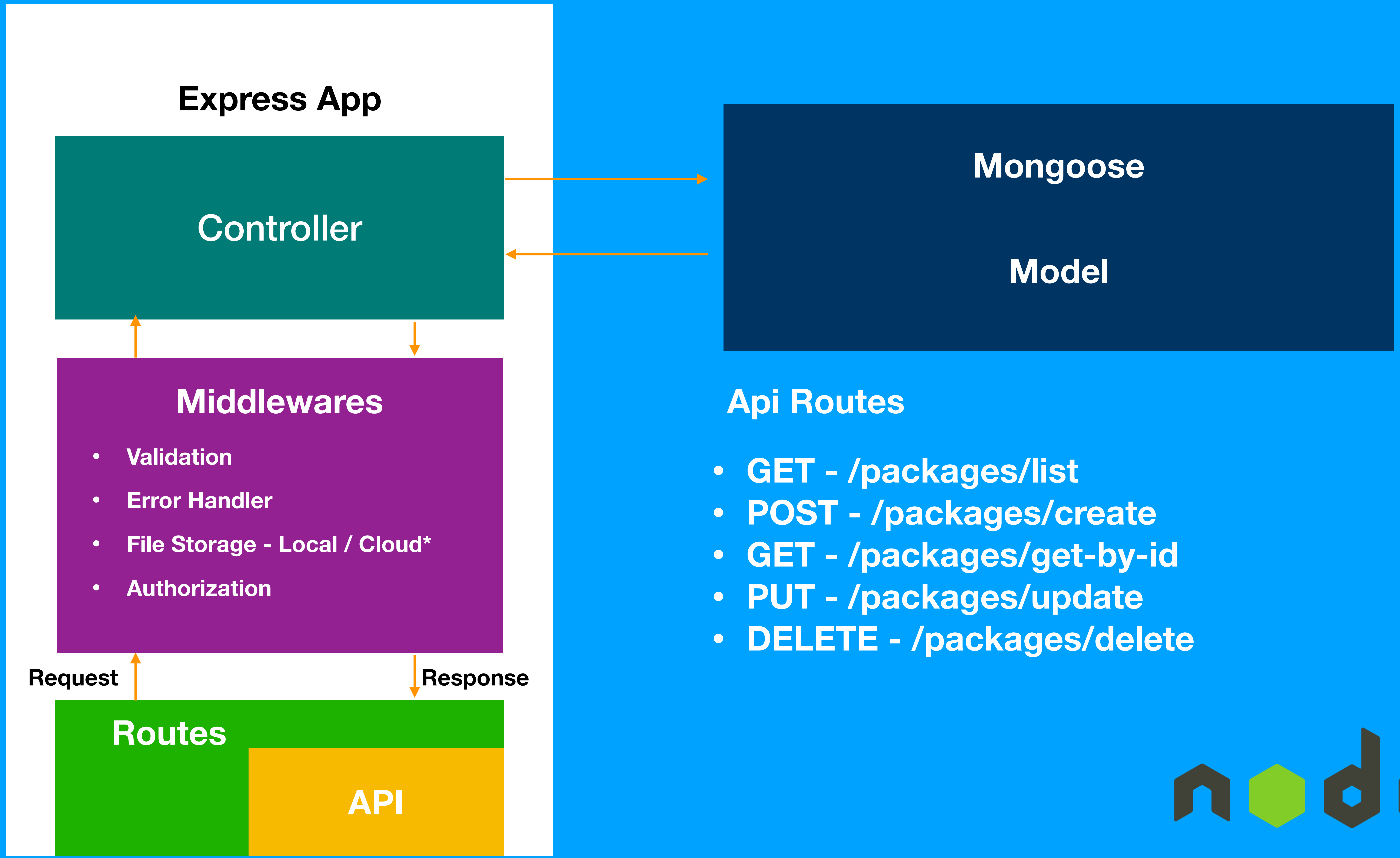
Model View Controller - (MVC)




Express App
Controller
Routes
API
Mongoose
Model
Datenbank
mongoDB
node JS
Get, Post, Put, Delete
Service / Model
Notify
Update
Controller
Update
Action
View
HTML - CSS

Backend

REST-API

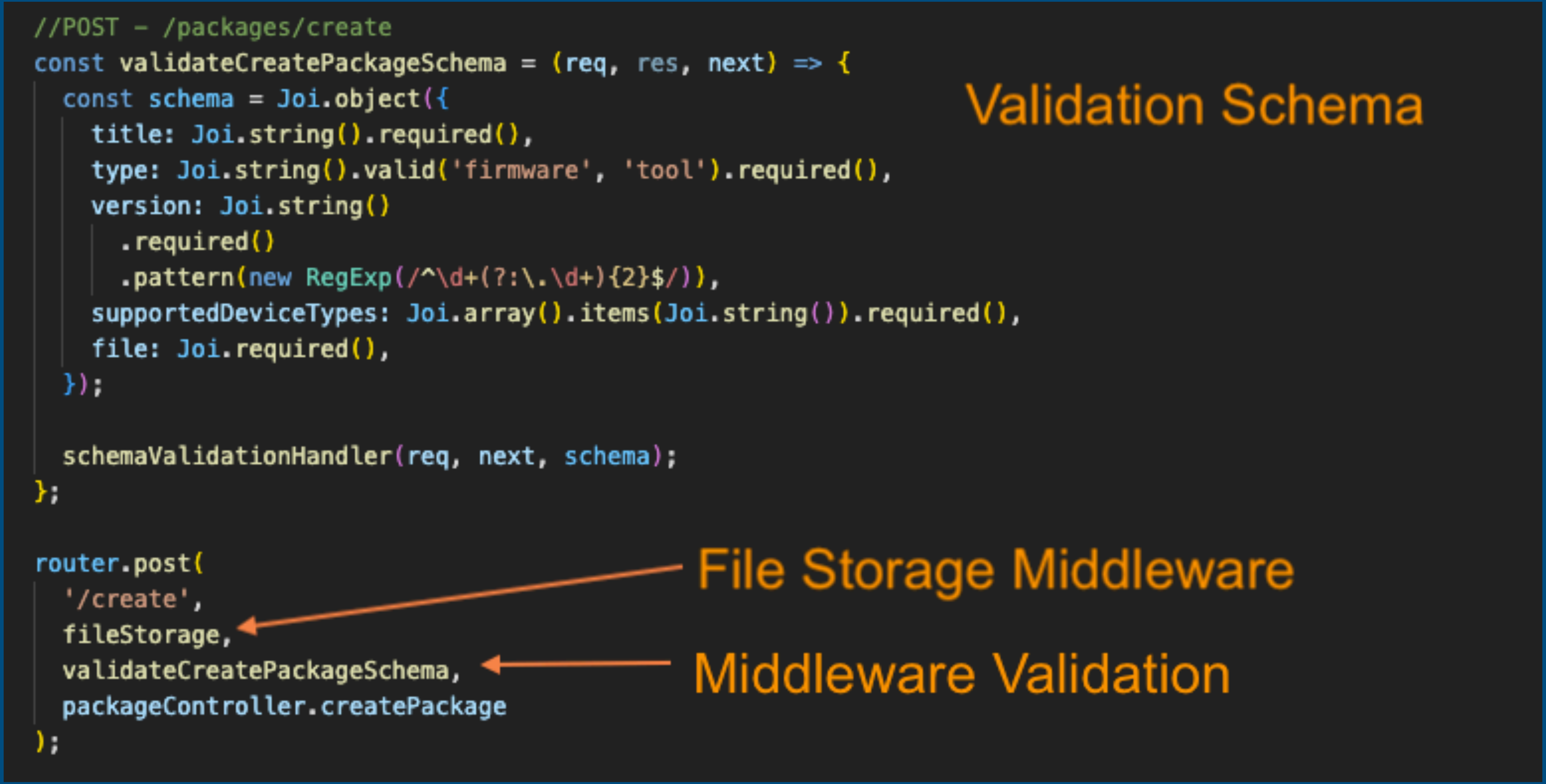


Api Routes

- GET - /packages/list
- POST - /packages/create
- GET - /packages/get-by-id
- PUT - /packages/update
- DELETE - /packages/delete

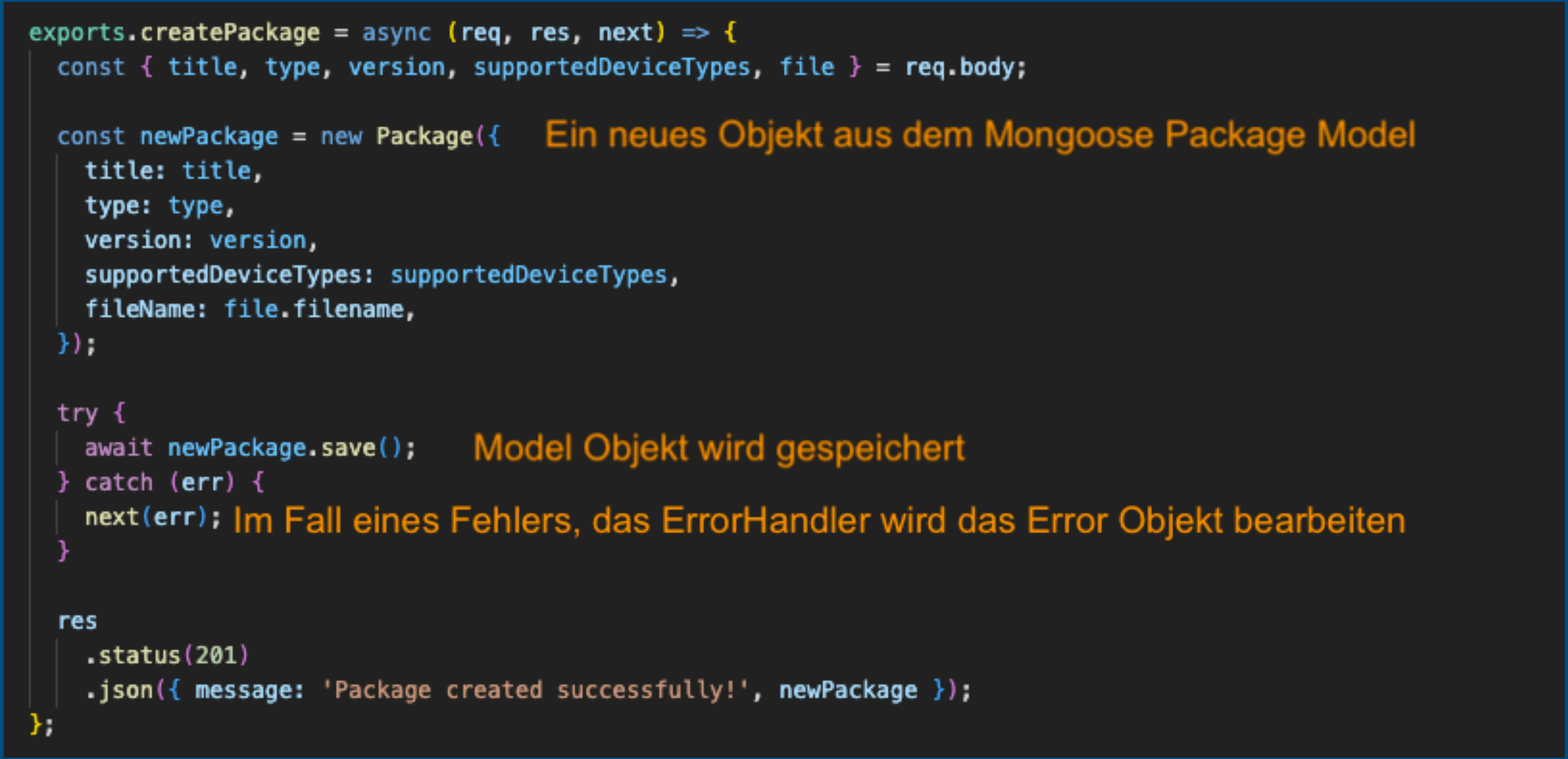
*In diesem Projekt nicht implementiert

Routes

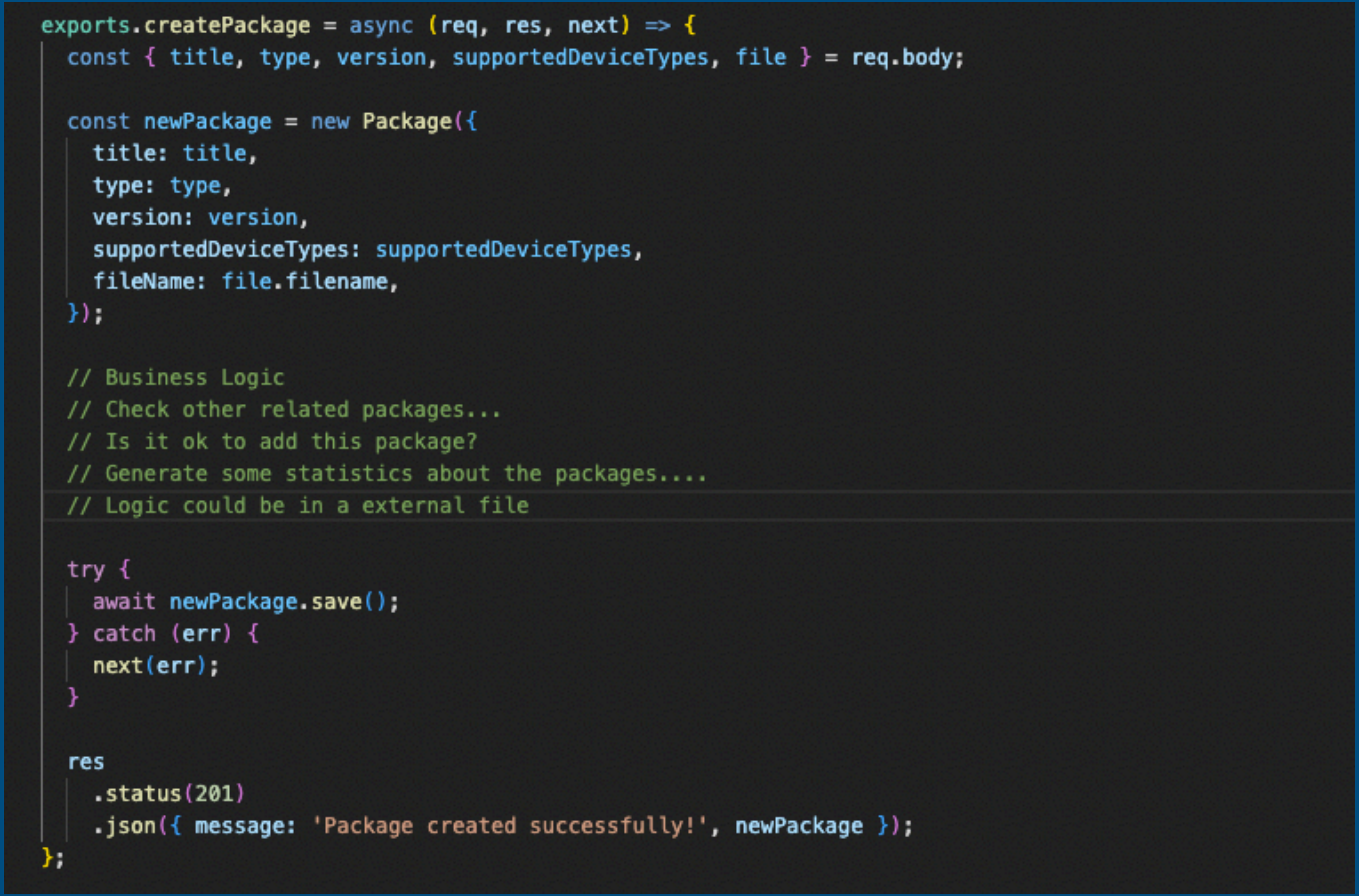


```
//POST - /packages/create
const validateCreatePackageSchema = (req, res, next) => {
  const schema = Joi.object({
    title: Joi.string().required(),
    type: Joi.string().valid('firmware', 'tool').required(),
    version: Joi.string()
      .required()
      .pattern(new RegExp(/^\d+(?:\.\d+){2}$/)),
    supportedDeviceTypes: Joi.array().items(Joi.string()).required(),
    file: Joi.required(),
  });

  schemaValidationHandler(req, next, schema);
};

router.post(
  '/create',
  fileStorage,
  validateCreatePackageSchema,
  packageController.createPackage
);
```

Controller

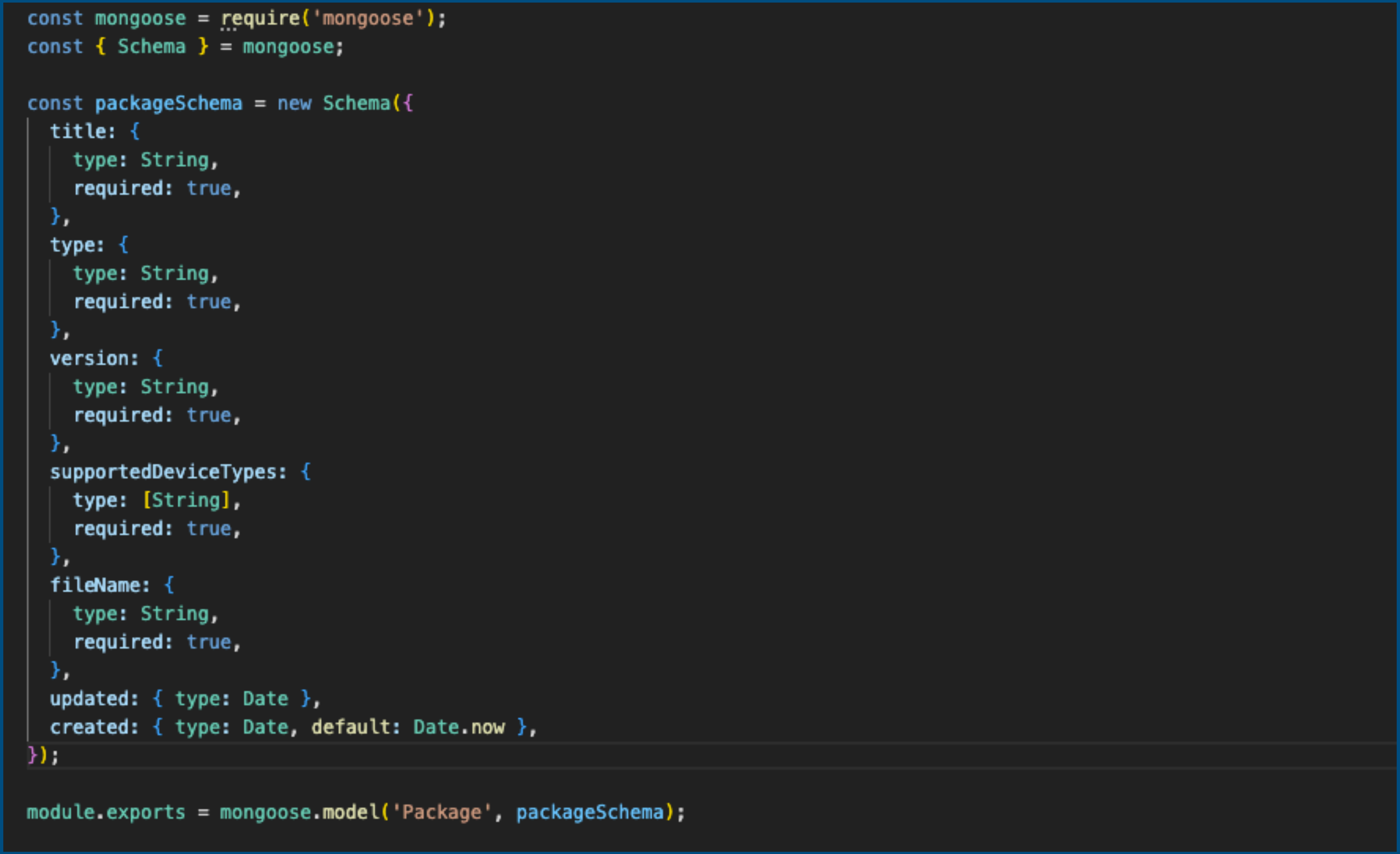


```
exports.createPackage = async (req, res, next) => {
  const { title, type, version, supportedDeviceTypes, file } = req.body;

  const newPackage = new Package({   Ein neues Objekt aus dem Mongoose Package Model
    title: title,
    type: type,
    version: version,
    supportedDeviceTypes: supportedDeviceTypes,
    fileName: file.filename,
  });

  try {
    await newPackage.save();   Model Objekt wird gespeichert
  } catch (err) {
    next(err); Im Fall eines Fehlers, das ErrorHandler wird das Error Objekt bearbeiten
  }

  res
    .status(201)
    .json({ message: 'Package created successfully!', newPackage });
};
```



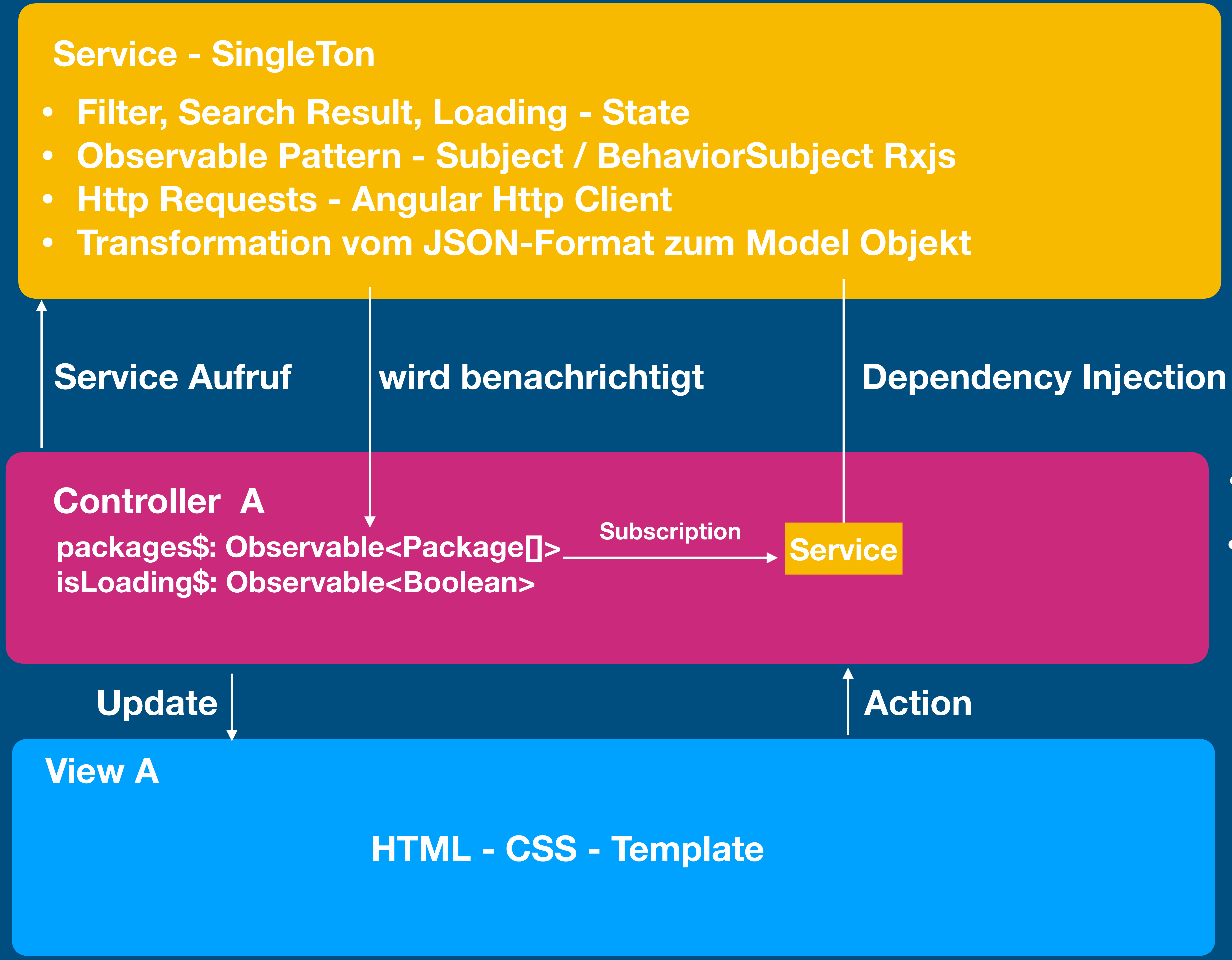
```
exports.createPackage = async (req, res, next) => {
  const { title, type, version, supportedDeviceTypes, file } = req.body;

  const newPackage = new Package({
    title: title,
    type: type,
    version: version,
    supportedDeviceTypes: supportedDeviceTypes,
    fileName: file.filename,
  });

  // Business Logic
  // Check other related packages...
  // Is it ok to add this package?
  // Generate some statistics about the packages....
  // Logic could be in a external file

  try {
    await newPackage.save();
  } catch (err) {
    next(err);
  }

  res
    .status(201)
    .json({ message: 'Package created successfully!', newPackage });
};
```

Model



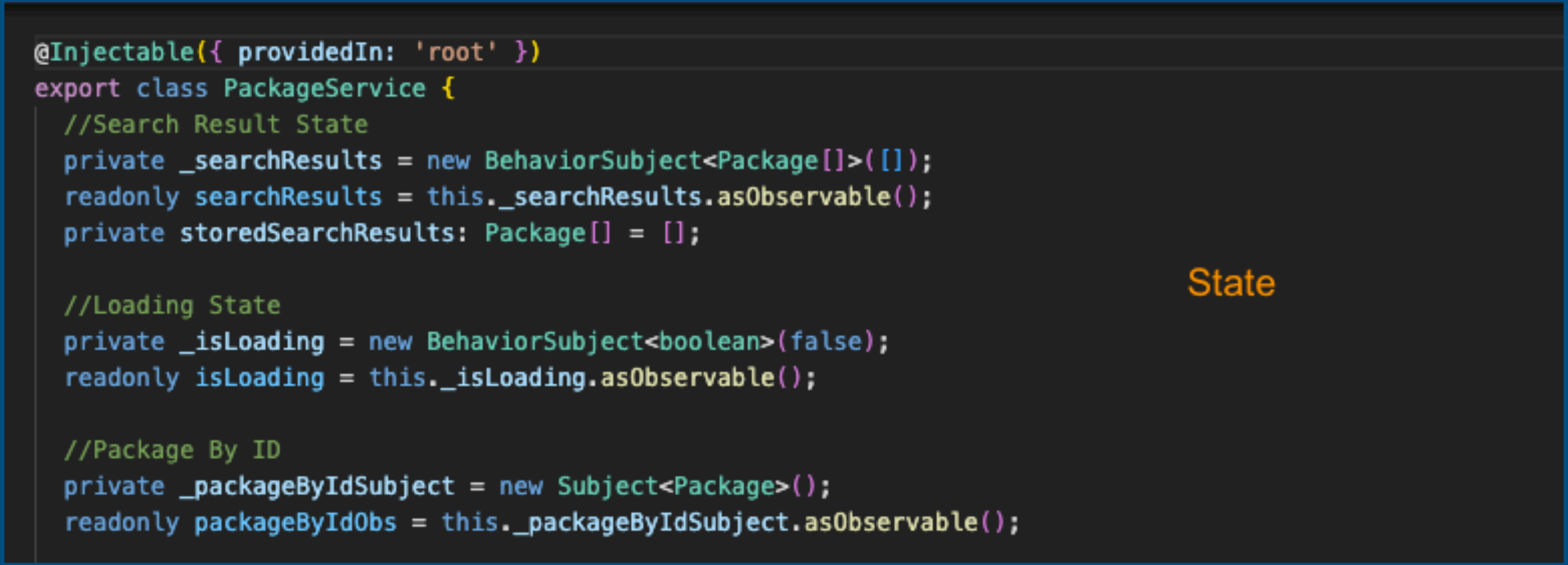
```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const packageSchema = new Schema({
  title: {
    type: String,
    required: true,
  },
  type: {
    type: String,
    required: true,
  },
  version: {
    type: String,
    required: true,
  },
  supportedDeviceTypes: {
    type: [String],
    required: true,
  },
  fileName: {
    type: String,
    required: true,
  },
  updated: { type: Date },
  created: { type: Date, default: Date.now },
});

module.exports = mongoose.model('Package', packageSchema);
```

Frontend


Service - SingleTon
• Filter, Search Result, Loading - State
• Observable Pattern - Subject / BehaviorSubject Rxjs
• Http Requests - Angular Http Client
• Transformation vom JSON-Format zum Model Objekt
Service Aufruf
wird benachrichtigt
Dependency Injection
Controller A
packages$: Observable<Package[]>
isLoading$: Observable<Boolean>
Subscription
Service
Update
Action
View A
HTML - CSS - Template

- Umgang mit Asynchronität durch rxjs Observables
- Bei größeren Projekten kann es sinnvoll sein, NGRX für State Management zu nutzen.

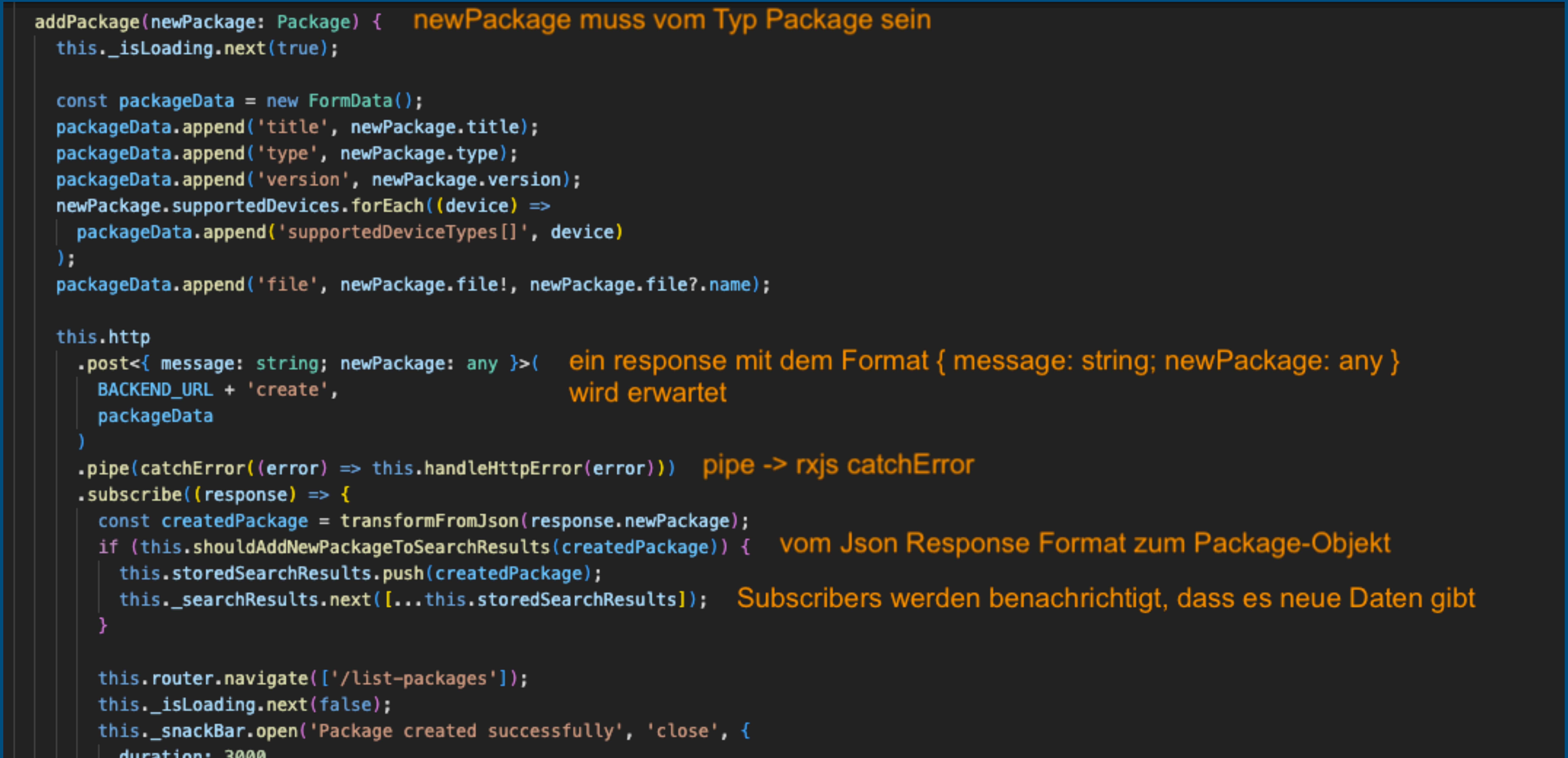
Service - State



```
@Injectable({ providedIn: 'root' })
export class PackageService {
  //Search Result State
  private _searchResults = new BehaviorSubject<Package[]>([]);
  readonly searchResults = this._searchResults.asObservable();
  private storedSearchResults: Package[] = [];

  //Loading State
  private _isLoading = new BehaviorSubject<boolean>(false);
  readonly isLoading = this._isLoading.asObservable();

  //Package By ID
  private _packageByIdSubject = new Subject<Package>();
  readonly packageByIdObs = this._packageByIdSubject.asObservable();
```

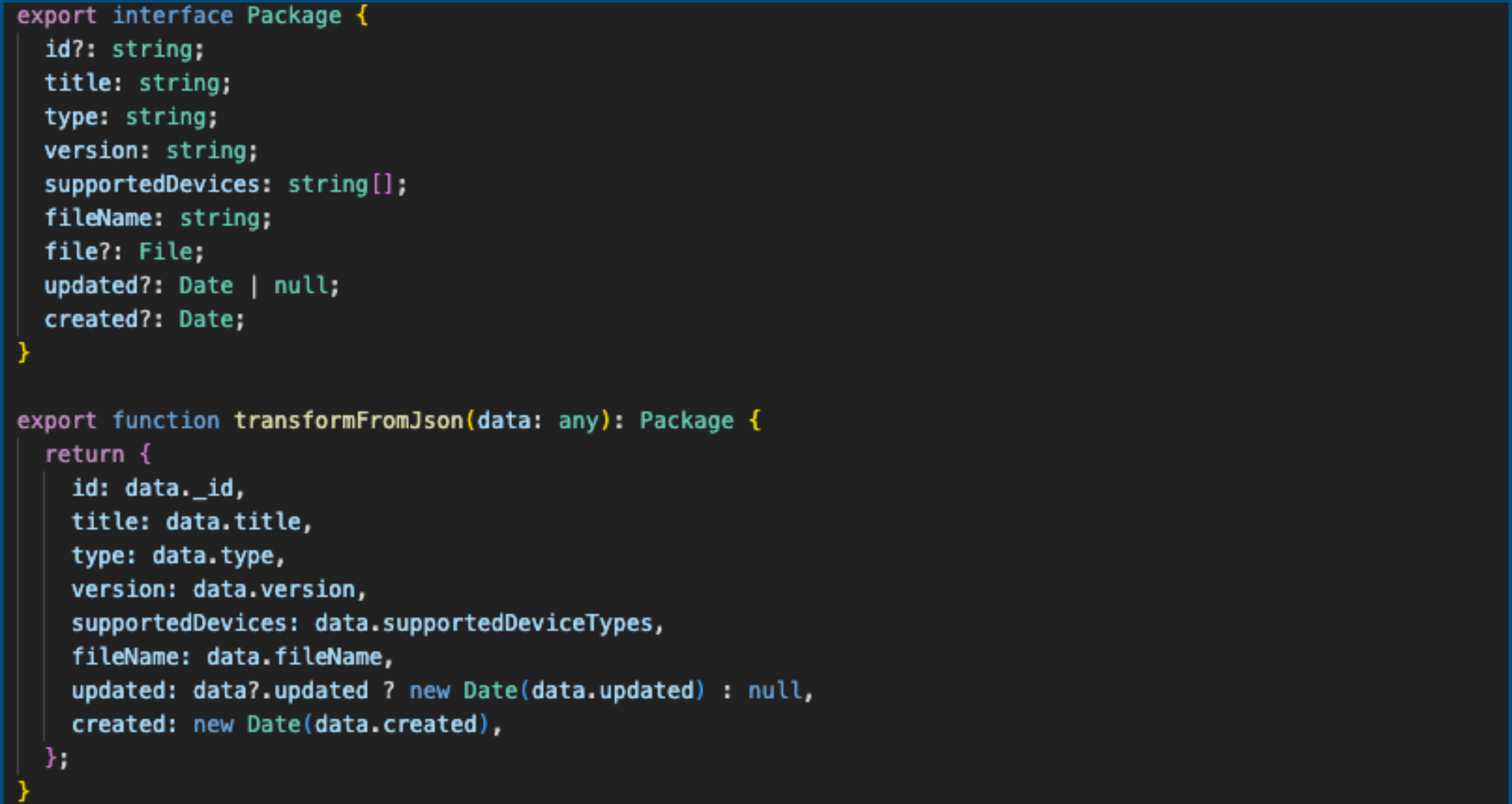
State

Service



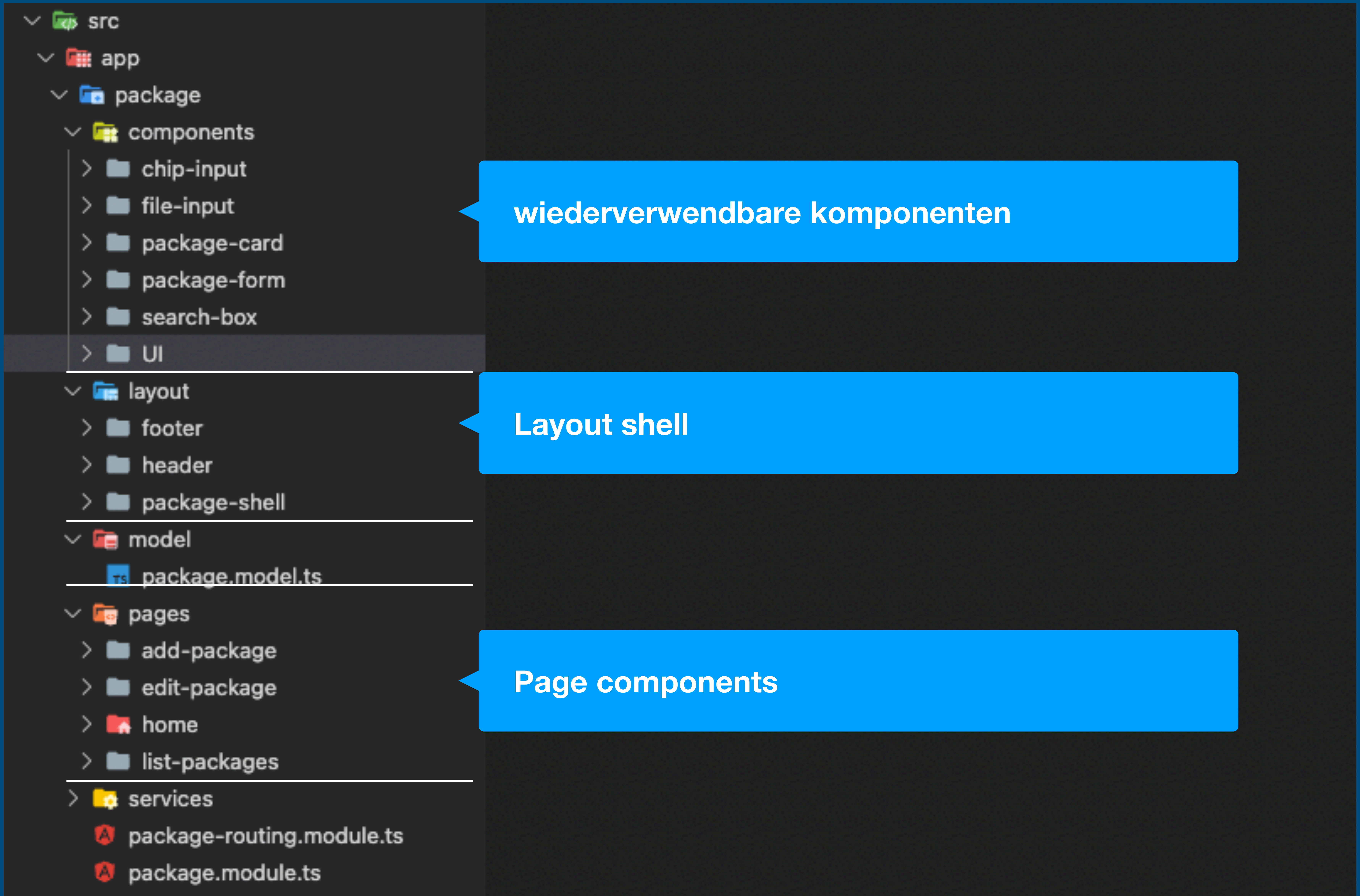
```
addPackage(newPackage: Package) {   newPackage muss vom Typ Package sein
  this._isLoading.next(true);

  const packageData = new FormData();
  packageData.append('title', newPackage.title);
  packageData.append('type', newPackage.type);
  packageData.append('version', newPackage.version);
  newPackage.supportedDevices.forEach((device) =>
    packageData.append('supportedDeviceTypes[]', device)
  );
  packageData.append('file', newPackage.file!, newPackage.file?.name);

  this.http
    .post<{ message: string; newPackage: any }>(   ein response mit dem Format { message: string; newPackage: any }
      BACKEND_URL + 'create',                       wird erwartet
      packageData
    )
    .pipe(catchError((error) => this.handleHttpError(error)))   pipe -> rxjs catchError
    .subscribe((response) => {
      const createdPackage = transformFromJson(response.newPackage);
      if (this.shouldAddNewPackageToSearchResults(createdPackage)) {   vom Json Response Format zum Package-Objekt
        this.storedSearchResults.push(createdPackage);
        this._searchResults.next([...this.storedSearchResults]);   Subscribers werden benachrichtigt, dass es neue Daten gibt
      }

      this.router.navigate(['/list-packages']);
      this._isLoading.next(false);
      this._snackBar.open('Package created successfully', 'close', {
        duration: 3000
```

Model - Interface



```
export interface Package {
  id?: string;
  title: string;
  type: string;
  version: string;
  supportedDevices: string[];
  fileName: string;
  file?: File;
  updated?: Date | null;
  created?: Date;
}

export function transformFromJson(data: any): Package {
  return {
    id: data._id,
    title: data.title,
    type: data.type,
    version: data.version,
    supportedDevices: data.supportedDeviceTypes,
    fileName: data.fileName,
    updated: data?.updated ? new Date(data.updated) : null,
    created: new Date(data.created),
  };
}
```

Components



```
src
  app
    package
      components
        chip-input
        file-input
        package-card
        package-form
        search-box
        UI
      layout
        footer
        header
        package-shell
      model
        package.model.ts
      pages
        add-package
        edit-package
        home
        list-packages
      services
      package-routing.module.ts
      package.module.ts
```

wiederverwendbare komponenten

Layout shell

Page components