

RENT A CAR APP Documentation.  
By Vasil Kostadinov 1901681053 - STD3

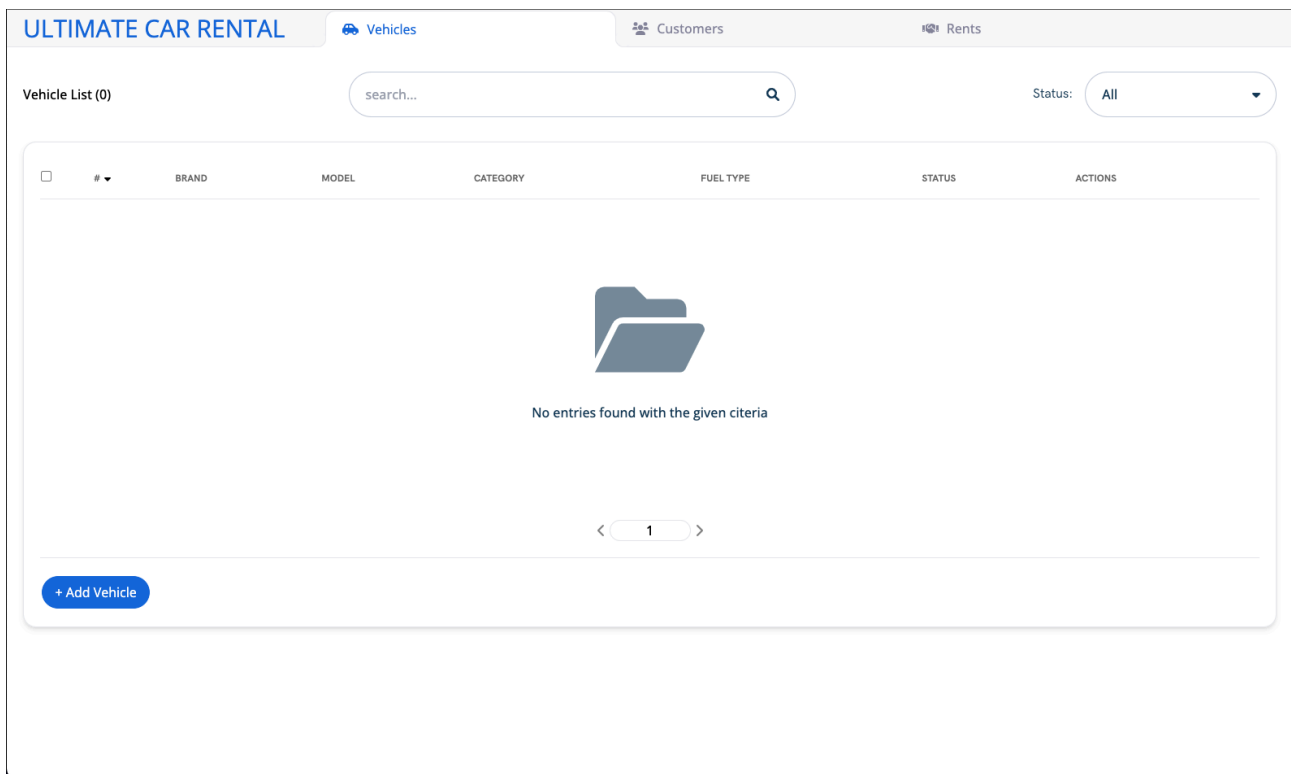
Tech used:

- NEXT Js
- json-server

Additional dependencies:

- "@fortawesome/fontawesome-svg-core": "^6.1.1",
- "@fortawesome/free-solid-svg-icons": "^6.1.1",
- "@fortawesome/react-fontawesome": "^0.1.18",
- "@tailwindcss/line-clamp": "^0.2.1",
- "@types/js-cookie": "^2.2.7",
- "@types/json-server": "^0.14.4",
- "axios": "^0.21.4",
- "framer-motion": "^6.2.8",
- "js-cookie": "^3.0.1",
- "json-server": "^0.17.0",
- "next": "^12.1.4",
- "next-images": "^1.7.0",
- "next-share": "^0.12.2",
- "next-sitemap": "^1.6.168",
- "npm": "^8.6.0",
- "postcss-flexbugs-fixes": "^5.0.2",
- "postcss-preset-env": "^7.4.1",
- "process": "^0.11.10",
- "react": "18.0.0",
- "react-dom": "18.0.0",
- "react-hook-form": "^7.17.1",
- "react-line-awesome": "^1.3.2",
- "react-loader-spinner": "4.0.0",
- "react-toastify": "^8.2.0",
- "tailwindcss": "^3.0.23",
- "tailwindcss-gradients": "^3.0.0",
- "yarn": "^1.22.18"
- },
- "devDependencies": {
- "@types/node": "17.0.13",
- "@types/react": "17.0.38",
- "autoprefixer": "^10.3.4",
- "css-loader": "^6.2.0",
- "eslint": "8.12.0",
- "eslint-config-next": "12.1.4",
- "eslint-plugin-react-hooks": "^4.4.0",
- "postcss": "^8.4.5",
- "postcss-cli": "^9.1.0",
- "style-loader": "^3.2.1",
- "typescript": "^4.6.3"
- }

## APP LAYOUT



### Navbar:

- Has 3 tabs, each leading to the 3 given collections: Vehicles, Customers and Rents

### Main:

- Count of given collection
- Search bar, allows you to search by every field in the collection (case insensitive, cannot correct errors)
- Filter, filters rows by different criteria

### Table Head:

- Select all visible entries checkbox
- Field names, every which of them is clickable and sorts the rows by that field. The arrow next to the name of the field indicates that the table is sorted by this field, and the direction of the arrow tells which order (asc or desc). Defaults to #V (id:desc , or newest first)

ULTIMATE CAR RENTAL

Vehicles


























Customers

Rents

Vehicle List (8)

search...

Status: All

<input type="checkbox"/>	#	BRAND	MODEL	CATEGORY	FUEL TYPE	STATUS	ACTIONS
<input checked="" type="checkbox"/>	8	Test vehicle	2	SUV	petrol	Rented	  
<input type="checkbox"/>	7	Test vehicle		SUV	petrol	Free	  
<input type="checkbox"/>	6	Test vehicle		SUV	petrol	Free	  
<input type="checkbox"/>	5	Test vehicle	2	SUV	petrol	Free	  
<input type="checkbox"/>	4	Test vehicle	1	economy	petrol	Free	  
<input type="checkbox"/>	3	Test vehicle	1	economy	petrol	Free	  
<input type="checkbox"/>	2	Test vehicle	1	economy	petrol	Free	  
<input type="checkbox"/>	1	Test vehicle	1	economy	petrol	Free	  

< 1 >

+ Add Vehicle

Delete selected

### Main table:

- Checkbox that allows us to select multiple rows and perform mass actions on them
- Field values
- Hovering on a row shows the image of the given entry, if it has one
- Action buttons for copying, editing or deleting a row (soft delete is not implemented, the entry is deleted forever)
- Pagination with back and next arrows and an input field that shows the current page, it is editable. A page number may be entered and on enter click or on blur, if that page exists, it is going to be displayed
- Add entry button, to add new entries to the collection
- Mass actions delete button, deletes all selected entries

×

Edit Vehicle

Brand \*

Test vehicle

Model \*

2

Category \*

SUV

Fuel type \*

petrol

Number of Seats \*

6

Construction year \*

2012


Price per day (\$) \*

150

Image: (url) \*

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRnKNkq-

Edit Vehicle



#### Creation/Edit Modal:

- Title that changes to edit if in edit mode
- Validated inputs (text, number, select boxes) That glow red if invalidated
- Custom select boxes that show the selectable options for each collection
- Image preview (if an url is pasted, it displays the picture, otherwise it stays blank)
- Add/update button

\*Note: Rented vehicles cannot be edited while in rent

×

Add vehicle

Brand \*

Chevrolet

Model \*

Corvette

Category \*

Economy

Fuel type \*

Petrol

Number of Seats \*

This field is required!

Construction year \*

This field is required!


Price per day (\$) \*

This field is required!

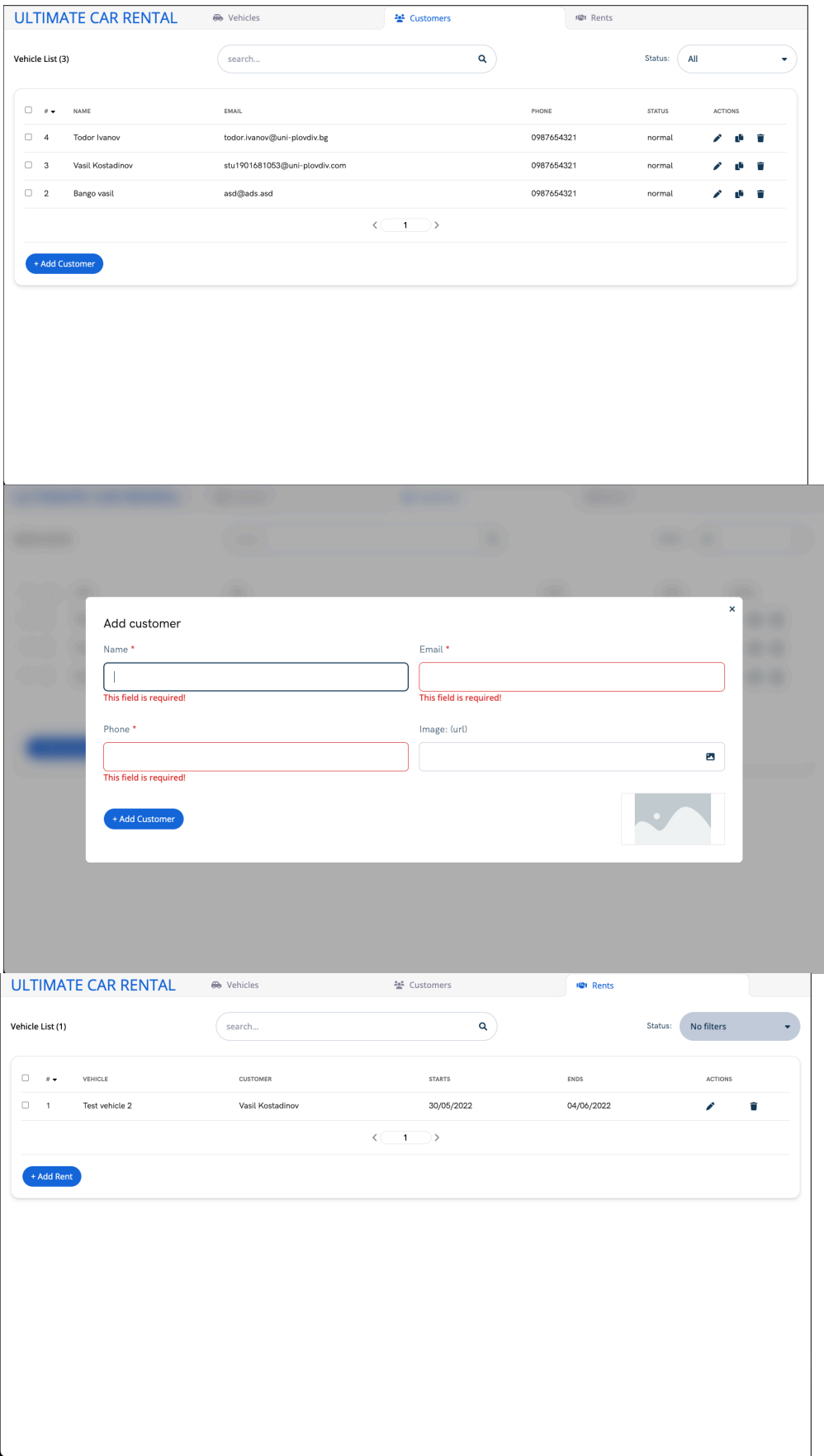
Image: (url) \*

data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAQ/2wCEA/



+ Add Vehicle



The other collections and previews are identical



Add Rent

Customer \*

Vehicle \*



Start Date

End Date

+ Add Rent

Days: 0, Price: \$0

Add Rent

Customer \*

Vehicle \*

Start Date

End Date

+ Add Rent

Days: 4, Price: Discount: 5% \$456

## RENT CREATION

- 2 dropdown menus that have the existing customers and the UNrented vehicles as options
- Start date and end date datepickers (default html inputs)
- a calculator that shows the number of days and the price (days \* price per day of the vehicle entry). It also shows the percentage discount if a vehicle is rented for more days.
- If a person is VIP, a crown will be shown next to their name, indicating that they are eligible for a 15% discount (all other discounts are invalid)
- Add button

## CODE BASE

interfaces:

```
namespace Vehicle {
  interface IVehicle {
    id?: number
    category?: StaticTypes.VehicleCategory
    fuelType?: StaticTypes.FuelType
    brand?: string
    model?: string
    constructionYear?: number
    numberOfSeats?: StaticTypes.SeatsNumber
    image?: string
    rented?: boolean
    pricePerDay?: number
    /**has many rents */
    rents?: Array<number> & Array<Rent.IRent>
    createdAt?: Date
    updatedAt?: Date | null
    description?: string
  }

  type VehicleInput = {
    category?: StaticTypes.VehicleCategory
    fuelType?: StaticTypes.FuelType
    brand?: string
    model?: string
    constructionYear?: number
    numberOfSeats?: StaticTypes.SeatsNumber
    image?: string
    rented?: boolean
    pricePerDay?: number
    rents?: Array<number>
    createdAt?: Date
    updatedAt?: Date | null
  }
}

namespace Customer {
  interface ICustomer {
    id: number
    name: string
    email: string
    phone: string
    vip: boolean
    image?: string
    /**has many rents */
    rents: Array<number> & Array<Rent.IRent>
    createdAt: Date
    updatedAt: Date | null
  }

  type CustomerInput = {
    name?: string
    email?: string
    phone?: string
    vip?: boolean
    image?: string
    rents?: Array<number>
    createdAt?: Date
    updatedAt?: Date | null
  }
}

namespace Rent {
  interface IRent {
    id: number
    start: string
    end: string
    vehicle: number & Vehicle.IVehicle //id
    customer: number & Customer.ICustomer
    price: number
    createdAt: Date | string | null
    updatedAt: Date | null
  }

  type RentInput = {
    start: string
    end: string
    vehicle: number
    customer: number
    price: number
    createdAt?: Date
    updatedAt?: Date | null
  }
}

interface IDBManager {
  create(
    collection: StaticTypes.Collection,
    body: StaticTypes.Input
  ): Promise<StaticTypes.Output>
  update(
    id: number,
    collection: StaticTypes.Collection,
    body: StaticTypes.Input
  ): Promise<boolean>
  delete(id: number, collection: StaticTypes.Collection): Promise<boolean>
  getOne(
    id: number,
    collection: StaticTypes.Collection
  ): Promise<StaticTypes.Output>
  getMany(
    collection: StaticTypes.Collection,
    q?: string,
    sort?: string,
    order?: string,
    customField?: string
  ): Promise<Array<StaticTypes.Output>>
}
```

```

namespace StaticTypes {
  type VehicleCategory = 'economy' | 'estate' | 'luxury' | 'SUV' | 'cargo'
  type FuelType = 'petrol' | 'diesel' | 'hybrid' | 'electric' | 'LPG' | 'CNG'
  type SeatsNumber = 2 | 3 | 4 | 5 | 6 | 7 | 8
  type Collection = 'rents' | 'vehicles' | 'customers'
  type Input = Customer.CustomerInput | Rent.RentInput | Vehicle.VehicleInput
  type Output =
    | Vehicle.IVehicle
    | Customer.ICustomer
    | Rent.IRent
    | Vehicle.IVehicle[]
    | Customer.ICustomer[]
    | Rent.IRent[]
    | { id: null; error: string; code: number }
}

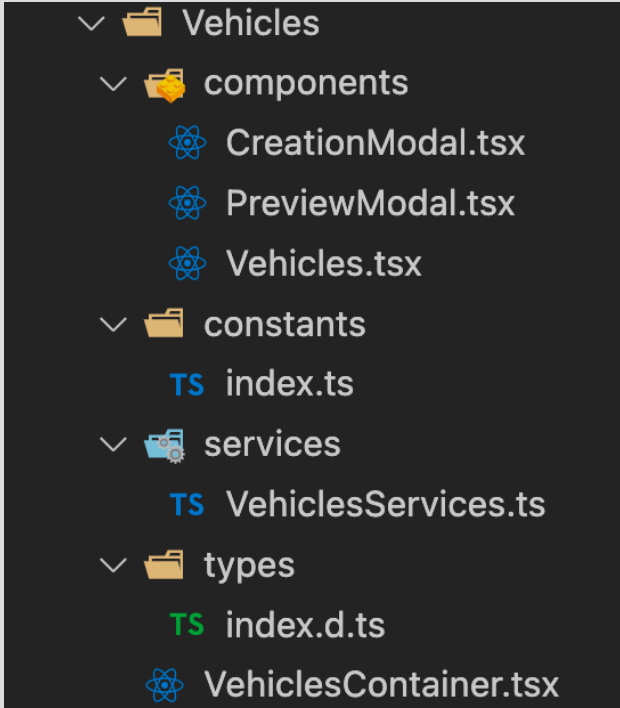
```

## PROJECT STRUCTURE

- public folder
- src folder
  - components : React Components used in the project
  - config : Project config constants
  - hooks : custom hooks
  - icons : svgs
  - modules : All Collection Tabs' codebases
  - pages : next routing
  - styles : tailwindcss
  - types : \*interfaces that are above this list\*
  - utils : global functions, used in the project
- .env
- eslint config
- gitignore
- prettier config
- db.json (database)
- next config
- package.json
- readme (this file)
- tailwind.config.js
- tsconfig.json
- lock files of package managers



## File convention:

 <pre>   ✓ Vehicles     ✓ components       CreationModal.tsx       PreviewModal.tsx       Vehicles.tsx     ✓ constants       TS index.ts     ✓ services       TS VehiclesServices.ts     ✓ types       TS index.d.ts       VehiclesContainer.tsx </pre>	<p>Container file: All state variables All queries Delegated functions returns: All presenters (controls, filters, table)</p> <p>types: module-specific types</p> <p>services: Functions that the container calls (moved outside of container file to save space)</p> <p>constants: constants used in the module</p> <p>components: The presenters, returned by the Container</p> <p>&lt;ModuleName&gt;.tsx is the table PreviewModal.tsx is when you click on the entry CreationModal.tsx the creation/edit form</p>

## Hooks:

- `useQuery(collection:Collection,sort:string,order:string): {data:Collection<T>,loading:bool,error:Error,refetch:RefetchFunc}`  
`refetch(q?: string, sort?: string, order?: string, customField?: string): void`  
fetches data using DBManager Class
- `useScroll():[allowScroll(),blockScroll()]`
  - blocks window scrolling
- `useWindowSize():size:{width:number,height:number}`
  - returns inner window size in pixels

## Utils:

- Cron
  - every one minute the cron executes a function
    - Update rented vehicles status( if a car is rented, update the status and vice versa)
    - Update customer VIP status. check if he has 3 rents in the last 60 days, if so, make him vip, if not, make him normal again