**Stages building a React app**

**סעיף 1 – יצירת פרויקט חדש – ללא ניקוד**

צרו פרויקט SPA חדש, התקינו dependencies ככל שתידרשו כפי שנלמד בכיתה

Basic

1. Create the basic app
   1. In the address bar of the folder you wish to place your project:

**cmd**

* 1. In the cmd create the project:

**create-react-app rick\_and\_morty --template typescript**

**cd rick\_and\_morty**

**code .**

1. Open development server – run your project – in a new terminal:

**npm start**

1. לנקות קצת... Clean the template of React:
   1. In App.tsx:
      * Delete the Header section and the unnecessary Import
   2. In App.css:
      * Delete all the unnecessary styling
2. Install necessary packages – in a different terminal:

**npm i react-router-dom @types/react-router-dom**

**npm i axios**

**npm i react-icons**

**npm i moment**

1. Create new components, as your project require – in the terminal:
   1. Layout components

**create fc Layout/Header**

**create fc Layout/Main**

**create fc Layout/Menu**

**create fc Layout/Footer**

**create fc Layout/Routing**

* 1. Pages

**create fc Pages/Home**

**create fc Pages/Page404**

**create fc Pages/RMCard**

**create fc Pages/RMTable**

**create fc Pages/About**

* 1. Shared

**create fc Shared/Card**

**create fc Shared/Table**

1. Insert the Layout components into the App component
   1. In App.tsx
   2. Make sure each component is imported

**<Header/>**

**<Menu/>**

**<Main/>**

**<Footer/>**

1. Format the App.css:
   1. Reset the style and create a grid:

**.App{**

**height: 100%;**

**width: 100%;**

**display: grid;**

**grid-template-rows: repeat(12, 1fr);**

**grid-template-columns: repeat(12, 1fr);**

**}**

* 1. Create the grid for all the components of the App:

**.App>.Header {**

**grid-column: 12 span;**

**grid-row: 1 span;**

**}**

**.App>.Menu {**

**grid-column: 2 span;**

**grid-row: 10 span;**

**}**

**.App>.Main {**

**grid-column: 10 span;**

**grid-row: 10 span;**

**}**

**.App>.Footer {**

**grid-column: 12 span;**

**grid-row: 1 span;**

**}**

1. In Index.css:
2. Import a font if you'd like

**@import url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap');**

**body {**

**font-family: Poppins, sans-serif;**

**-webkit-font-smoothing: antialiased;**

**-moz-osx-font-smoothing: grayscale;**

**}**

1. Reset the styles

**html, body, #root{**

**height: 100%;**

**width: 100%;**

**padding: 0px;**

**margin: 0px;**

**}**

1. Create basic classes that will be needed across the project

**.centerRow {**

**display: flex;**

**flex-direction: row;**

**justify-content: center;**

**align-items: center;**

**}**

**.centerColumn}**

**display: flex;**

**flex-direction: column;**

**justify-content: center;**

**align-items: center;**

**}**

* 1. Determine variables for colors, fonts etc (you can select color themes in colorhunt or cooler).

:**root {**

**--primary-color: #191970;**

**--background-color: #FCFCFC;**

**--secondary-color: #008080;**

**--font-small: 15px;**

**--font-medium: 30px;**

**--font-large: 40px;**

**}**

* 1. If you don't have a light/dark mode, set the color and background color

**body {**

**font-family: Poppins, sans-serif;**

**-webkit-font-smoothing: antialiased;**

**-moz-osx-font-smoothing: grayscale;**

**color: var(--primary-color);**

**background-color: var(--background-color);**

**font-size: var(--font-medium);**

**}**

**סעיף 3 – יצירת מנגנון Routing – 30 נקודות**

יצירת מנגנון Routing מתאים המאפשר לגלוש לדף הבית ולדף המדינות ע"י התפריט הראשי.

\*\* שימו לב – אני פותרת את סעיף 3 לפני סעיף 2 כיוון שבלעדיו לא אראה בדפדפן אף אחד מהשינויים שאערוך בסעיף 2.

Routing

1. Install react-router-dom library (the latest version):
   * + **npm i react-router-dom @types/react-router-dom**
2. Create content to the Home and Page404 components
   1. In Home.tsx:

**<div className="Home centerColumn">**

**<h2>Hi! Welcome to the Rick and Morty app.</h2>**

**<h2> We are so glad you've decided to join us.</h2>**

**<iframe**

**src="https://giphy.com/embed/cOKjNdJDbqNCm4n0Jm"**

**width="480"**

**height="480"**

**frameBorder="0"**

**className="giphy-embed"**

**allowFullScreen**

**title="Rick and Morty GIF">**

**</iframe>**

**</div>**

* 1. In index.css:

**h2{**

**font-size: var(--font-medium);**

**margin-bottom: 0.2rem;;**

**}**

* 1. In Page404.tsx:

**<div className="Page404 centerColumn">**

**<h2>Ho, no!</h2>**

**<h2>The page you were looking for does not exist!</h2>**

**<iframe**

**allow="fullscreen"**

**frameBorder="0"**

**width="480"**

**height="270"**

**src="https://giphy.com/embed/2N7E1MUmbz812T4TDk/video"**

**title="Rick and Morty GIF page404">**

**</iframe>**

**<p>How about going back to the <Link to="home">Home</Link> page?</p>**

**</div>**

1. Create content and style to the Menu component
   1. In Menu.tsx
   2. Make sure the Link is imported

**<div className="Menu">**

**<Link to="home">Home</Link>**

**<Link to="cards">Rick and Morty in Cards</Link>**

**<Link to="table">Rick and Morty in Table</Link>**

**<Link to="about">About</Link>**

**</div>**

1. Create content to the Routing component
   1. In Routing.tsx
   2. Make sure that the Routes and Route are imported
   3. Make sure every component is imported.
   4. Make sure you have an index (for Home) and an \* (for Page404) elements.
   5. The name of the page in the path needs to be identical to the one in the menu component (in the Link):

**<div className="Routing">**

**<Routes>**

**<Route path="/" element={<App/>}/>**

**<Route path="home" element={<Home />}/>**

**<Route index element={<Home />}/>**

**<Route path="about" element={<About/>}/>**

**<Route path="cards" element={<RMCard/>}/>**

**<Route path="table" element={<RMTable/>}/>**

**<Route path="\*" element={<Page404/>}/>**

**</Routes>**

**</div>**

1. Support URL & Components in index.tsx
   1. Add and import <BrowserRouter>
   2. Comment <React.StrictMode> for later

**ReactDOM.render(**

**//<React.StrictMode>**

**<BrowserRouter>**

**<App />**

**</BrowserRouter>**

**//</React.StrictMode>**

**document.getElementById('root') );**

1. Create content to the Main component
   1. In Main.tsx
   2. Make sure Routing and Outlet are imported

**<div className="Main">**

**<Routing/>**

**<Outlet/>**

**</div>**

* 1. Style the Main component, including treating the Routing (i.e., son) component in Main.css:

**.Main {**

**height:100%;**

**width: 100%;**

**overflow-y: auto;**

**}**

**.Routing{**

**height:100%;**

**}**

1. Insert the Layout component to App.tsx and make sure each component is imported in the top of the page:
   * + **<div className="App">**

**<Header />**

**<Menu />**

**<Main/>**

**<Footer/>**

**</div>**

**סעיף 2 – הסדרת Layout – 10 נקודות**

צרו את ה-Layout הכללי בקומפוננט App שיכיל רכיבים כמו Header, Menu, Main

אין צורך ליצור Footer.

* יצירת Component Header המכילה את כותרת האתר ותפריט
* יצירת Component Main שתציג תוכן מתחלף
* יצירת Component Menu שתציג את אפשרויות הניווט באתר
* יצירת About Component שתציג מידע כללי על האתר (2-3 שורות)

Layout

1. Creating a header component the includes headline and a small menu:
   1. Creating the content in Header.tsx:
   2. Make sure the Link is imported

**<div className="Header">**

**<h1>Rick and Morty app</h1<**

**</div>**

* 1. General style in index.css:

**.** **a, a:visited, a:active{**

**color: var(--primary-color);**

**text-decoration: none;**

**font-weight: 700;**

**margin: 1rem;**

**}**

**a:hover{**

**/\* text-decoration-line: underline; \*/**

**text-shadow: 5px 10px 18px var(--secondary-color);**

**font-style: oblique;**

**}**

1. Creating a Main component that will present a changing content – see Routing above.
2. Creating a Menu component
   1. In Menu.tsx:

**<div className="Menu">**

**<div className="centerColumn menuList">**

**<Link to="home">Home</Link>**

**<Link to="cards">Rick and Morty in Cards</Link>**

**<Link to="table">Rick and Morty in Table</Link>**

**<Link to="about">About</Link>**

**</div>**

**</div>**

1. Creating a Home page and Page404 - content – see Routing above.
2. Creating an About component
   1. In About.tsx:

**import "./About.css";**

**function About(): JSX.Element {**

**return (**

**<div className="About">**

**<h2>This is the Rick and Morty app</h2>**

**<p>Here you can see the different characters and read their properties: name, status (alive or unknown) species (human or alien).</p>**

**<p>You can also search according to status or species.</p>**

**<p>Hope you enjoy. </p>**

**</div>**

* 1. In About.css:

**.About {**

**text-align: left;**

**margin: 2rem;**

**}**

**סעיף 4 – משיכת מידע משרת מרוחק והצגתו באתר – 30 נקודות**

לצורך הבאת המידע, יש לגלוש ב-AJAX לכתובת הבאה, המחזירה JSON של כל המדינות: <https://rickandmortyapi.com/api/character>

הציגו את המידע בתצורת טבלה או כרטיסיות, הציגו את המידע הבא:

1. id – מזהה
2. name – שם הדמות
3. status – מצב הדמות (Alive, Unknown)
4. species – סוג דמות (Human, Align)
5. image – תמונת הדמות (רוחב 80 פיקסלים, גובה 80 פיקסלים).

השמות הללו אלו בדיוק השמות המופיעים ב-JSON המוחזר.

\*\* אני יצרתי גם כרטיסיות וגם טבלה בשביל התרגול.

Networking

1. Import axios

**npm i axios**

1. Create the interface of the data set:
   1. Copy the data set from the server
   2. Find an on line website that transform JSON to typescript and paste the data there. This will create an interface of the data set, i.e. a model of the way the data is build.
   3. In VScode, in the src folder, create a new folder name Models (right click on the folder, select the "new folder" option and write the name Models).
   4. In VScode, in the Models folder, create a new ts file and name it RickMortyModel.ts (right click on the folder, select the "new file" option and write the name you want with a postfix of ts).
   5. Copy the interface from the website (with the two upper rows with the Root) to the ts file.
   6. Note that in the current case – the data comes in the form of Root that contains info and reults array. Therefore we need to copy all the json model and later on we should call the result from the Root. Just follow the instructions.

**export interface Root {**

**info: Info**

**results: Result[]**

**}**

**export interface Info {**

**count: number**

**pages: number**

**next: string**

**prev: any**

**}**

**export interface Result {**

**id: number**

**name: string**

**status: string**

**species: string**

**type: string**

**gender: string**

**origin: Origin**

**location: Location**

**image: string**

**episode: string[]**

**url: string**

**created: string**

**}**

**export interface Origin {**

**name: string**

**url: string**

**}**

**export interface Location {**

**name: string**

**url: string**

**}**

1. Create a component that holds the data from the data set in the form of cards and another to hold the card of each item, and another to hold the data in the form of a table and another to hold the table.
   1. Create a relevant component in the Pages folder

**create fc Pages/RMCard**

**create fc Pages/RMTable**

* 1. Create a relevant component in the Shared folder

**create fc Shared/Card**

**create fc Shared/Table**

1. Create a prop for the data in the card component and insert the data into the component itself
   1. In the card component, Card.tsx
   2. Before the function create an interface that accept the CountryModel. from the CountryModel.ts file and make sure that the model is imported

**import { Result } from "../../../Models/RickMortyModel";**

**import "./Card.css";**

**interface CardProps{**

**rickMorty: Result;**

**}**

* 1. After the fonction, insert the model into the card component, including the img, and make sure to pass the props as an argument for the function

**function Card(props: CardProps): JSX.Element {**

**return (**

**<div className="Card centerColumn border">**

**<h3>{props.rickMorty.name}</h3>**

**<p><span className="headline">ID:</span> {props.rickMorty.id}</p>**

**<p><span className="headline">Status:</span> {props.rickMorty.status}</p>**

**<p><span className="headline">Specie:</span> {props.rickMorty.species}</p>**

**<img src={props.rickMorty.image} alt="Character" className="border"/>**

**</div>**

**);**

**}**

1. Format the card component.
   1. In Card.css.

**.Card {**

**margin: 1rem;**

**padding: 1rem;**

**}**

**.Card p{**

**font-size: var(--font-small);**

**margin: 0.3rem;**

**}**

* 1. In Index.css:

**.headline{**

**font-weight: 500;**

**text-decoration: underline;**

**}**

**.border{**

**border: 1px solid var(--secondary-color);**

**border-radius: 10px;**

**box-shadow: 5px 10px 18px var(--secondary-color);**

**}**

**h3{**

**font-size: var(--font-medium);**

**font-weight: 500;**

**margin: 0.5rem;**

**}**

**img{**

**width: 80px;**

**height: 80px;**

**}**

1. Import the data and insert it into the Component in cards form:
   1. In the parent component, RMCard.tsx
   2. After the function and before the return – create a state and import the data from the online json file.
   3. The state initialized with an empty array.
   4. Make sure to close the useEffect hook with []
   5. Make sure that the model is imported from the ts file.
   6. Make sure the hooks and the axios are imported.
   7. Make sure that in the axios get method, you specify the type of data imported, that is Root, and later to call specifically the result from the Root.

**import { useEffect, useState } from "react";**

**import "./RMCard.css";**

**import axios from "axios";**

**import Card from "../../Shared/Card/Card";**

**import { Result, Root } from "../../../Models/RickMortyModel";**

**function RMCard(): JSX.Element {**

**const [cards, setCards] = useState<Result[]>([]);**

**useEffect(() => {**

**axios.get<Root>('https://rickandmortyapi.com/api/character')**

**.then(res => {**

**setCards(res.data.results);**

**console.log('Data load successfully');**

**})**

**.catch(err => {console.log('Data did not load');})**

**}, []);**

* 1. After the return – call the card component
  2. Make sure to write the key

**return (**

**<div className="RMCard">**

**<h2>Rick and Morty characters</h2>**

**<div className="centerRow">**

**{**

**cards.map(c => <Card key={c.id} rickMorty={c} />)**

**}**

**</div>**

**</div>**

**);**

**}**

* 1. NOTE: this component will change in the next section.

1. Format the component:
   1. In the RMCard.css:

**.RMCard {**

**}**

**.RMCard .centerRow{**

**flex-wrap: wrap;**

**}**

1. Same thing for the RMTable: create a prop for the data in the table component and insert the data into the component itself
   1. In the table component, Table.tsx
   2. Before the function create an interface that accept the Result[] from the RickMortyModel.ts file and make sure that the model is imported.
   3. Note that here we import the data in the form of an array and that we import the Result array that is nested within the Root interface. This complexity is due to the organization of the data.

**import { Result } from "../../../Models/RickMortyModel";**

**import "./Table.css";**

**interface TableProps{**

**table: Result[];**

**}**

* 1. After the function, insert the model into the table component, including the img, and make sure to pass the props as an argument for the function

**function Table(props: TableProps): JSX.Element {**

**const header = ['Name', 'ID', 'Status', 'Species', 'Image'];**

**return (**

**<div className="Table">**

**<table>**

**<thead>**

**<tr>**

**{header.map(h => <th key={h}>{h}</th>)}**

**</tr>**

**</thead>**

**<tbody>**

**{props.table.map(t =>**

**<tr key={t.name}>**

**<td>{t.name}</td>**

**<td>{t.id}</td>**

**<td>{t.status}</td>**

**<td>{t.species}</td>**

**<td>**

**<img src={t.image} alt={`${t.name}'s picture`} />**

**</td>**

**</tr>**

**)**

**}**

**</tbody>**

**</table>**

**</div>**

**);**

1. Format the table component.
   1. In CountryTable.css:

**.Table {**

**font-size: var(--font-small);**

**}**

**table td, table th {**

**border: 1px solid var(--secondary-color);**

**padding: 8px;**

**}**

**table th{**

**text-align: left;**

**background-color: var(--secondary-color);**

**color: var(--background-color);**

**}**

1. Import the data and insert it into the Table component.
   1. In the parent component, RMTable.tsx
   2. After the function and before the return – create a state and import the data from the online json file.
   3. The state initialized with an empty array.
   4. Make sure to close the useEffect hook with []
   5. Make sure that the model is imported from the ts file.
   6. Make sure the hooks and the axios are imported.
   7. NOTE that although the data is in the Result array and that in the useState we specify that the format of the data is Result[], in the useEffect we still specify that the format of the imported data is Root, because our data is nested within it.

**import { useEffect, useState } from "react";**

**import { Result, Root } from "../../../Models/RickMortyModel";**

**import "./RMTable.css";**

**import axios from "axios";**

**import Table from "../../Shared/Table/Table";**

**function RMTable(): JSX.Element {**

**const [table, setTable] = useState<Result[]>([]);**

**useEffect(() => {**

**axios.get<Root>('https://rickandmortyapi.com/api/character')**

**.then(res => {**

**setTable(res.data.results);**

**console.log('Data load successfully');**

**})**

**.catch(err => {console.log('Data did not load');})**

**}, []);**

* 1. After the return – call the table component
  2. Note that here there is no need form map or key because the Table component in itself maps the data. We sent id the array.

**return (**

**<div className="RMTable">**

**<h2>Rick and Morty characters</h2>**

**{**

**<Table table={table}/>**

**}**

**</div>**

**);**

* 1. NOTE: this component will change in the next section.

**סעיף 5 – תמיכת חיפוש באתר – 30 נקודות**

**יצירת תיבת פילטור – הוסיפו תמיכה בפילטור המידע לפי status ו/או species**

Filtering

1. For the cards display add state for the search, search input and conditional rendering with the filter function
   1. In the RMCard.tsx
   2. After the function but before the return – add a state for the search

**const [selectStatus, setSelectStatus] = useState<string>('All');**

**const [selectSpecies, setSelectSpecies] = useState<string>('All');**

* 1. After the return – add an input and conditional rendering
  2. Note that for display purposes I've changed the h2 and put it inside a header

**return (**

**<div className="RMCard">**

**<div className="CardsHeader centerColumn">**

**<h2>Rick and Morty characters</h2>**

**<div className="centerRow">**

**<select placeholder="Select status" value={selectStatus} onChange={(e) => setSelectStatus(e.target.value)} className="border">**

**<option value="All">All</option>**

**<option value="Alive">Alive</option>**

**<option value="Dead">Dead</option>**

**<option value="unknown">Unknown</option>**

**</select>**

**<select placeholder="Select specie" value={selectSpecies} onChange={(e) => setSelectSpecies(e.target.value)} className="border">**

**<option value="All">All</option>**

**<option value="Human">Human</option>**

**<option value="Alien">Alien</option>**

**</select>**

**</div>**

**</div>**

**<div className="centerRow">**

**{**

**cards.filter((c) => {**

**const statusMatch = selectStatus === 'All' || c.status.toLowerCase() === selectStatus.toLowerCase();**

**const speciesMatch = selectSpecies === 'All' || c.species.toLowerCase() === selectSpecies.toLowerCase();**

**return statusMatch && speciesMatch;**

**}).map((c) => <Card key={c.id} rickMorty={c} />)**

**}**

**</div>**

**);**

**}**

1. Format the component:
   1. In RMCard.tsx:

**.RMCard {**

**}**

**.RMCard .centerRow{**

**flex-wrap: wrap;**

**}**

**.RMCard .CardsHeader{**

**background-color: var(--background-color);**

**color: inherit;**

**position: sticky;**

**top: 0;**

**z-index: 1;**

**}**

* 1. In Index.css:

**select{**

**font-family: inherit;**

**font-size: var(--font-small);**

**background-color: var(--background-color);**

**color: var(primary-color);**

**margin: 1rem;**

**}**

1. Similarly, for the RMTable. But note that the filter function is a little different from the cards, since the mapping of the array takes place inside the Table component and not the parent component, namely RMTable.
   1. In RMTable.tsx:
   2. After the function and before the return – add a state for the search

**const [selectStatus, setSelectStatus] = useState<string>('All');**

**const [selectSpecies, setSelectSpecies] = useState<string>('All');**

* 1. After the return – add a search input and conditional rendering
  2. Note that for display purposes I've changed the h2 and put it inside a header

**return (**

**<div className="RMTable">**

**<div className="CardsHeader centerColumn">**

**<h2>Rick and Morty characters</h2>**

**<div className="centerRow">**

**<div>**

**<select placeholder="Select status" value={selectStatus} onChange={(e) => setSelectStatus(e.target.value)} className="border">**

**<option value="All">All</option>**

**<option value="Alive">Alive</option>**

**<option value="Dead">Dead</option>**

**<option value="unknown">Unknown</option>**

**</select>**

**</div>**

**<div>**

**<select placeholder="Select specie" value={selectSpecies} onChange={(e) => setSelectSpecies(e.target.value)} className="border">**

**<option value="All">All</option>**

**<option value="Human">Human</option>**

**<option value="Alien">Alien</option>**

**</select>**

**</div>**

**</div>**

**</div>**

**{ <Table table={table.filter((t) => {**

**const statusMatch = selectStatus === 'All' || t.status.toLowerCase() === selectStatus.toLowerCase();**

**const speciesMatch = selectSpecies === 'All' || t.species.toLowerCase() === selectSpecies.toLowerCase();**

**return statusMatch && speciesMatch;**

**})}/>**

**}**

**</div>**

**);**

**}**