Лабораторная работа
«Создание многостраничного сайта – TypeScript, React и библиотека MUI »

Практическое задание. Используя TypeScript, React и библиотека MUI создать двухстраничный сайт, который включает следующие страницы:

- **Главная** (разработана в предыдущей лабораторной работе);
- **Список зданий**;
- **Диаграммы**.

На **Главной** странице по клику на рисунки галереи выводить страницу с описанием соответствующего здания или сооружения.

На странице **Список зданий** вывести список самых высоких зданий и сооружений в виде таблицы, реализовать пагинацию, фильтры и сортировку.

На странице **Диаграммы** вывести информацию о самых высоких зданиях и сооружениях в сгруппированном виде. Построить интерактивную диаграмму.

Задание 1. Создание проекта
Порядок выполнения практического задания

1. Создать проект **`practicum_MUI_pages.`** Скопировать его из **`practicum_MUI.`** Изменить структуру проекта. Для каждой страницы создать свою папку (**`main`** и **`list`**). Компоненты **`Navbar`** и **`Footer`** оставить в папке **`components`** проекта. В каждой папке для страниц создать свою папку **`components`**. Компоненты главной страницы (кроме **`Navbar`** и **`Footer`**) перенести в папку **`main/components/`**. Также туда перенести папку с рисунками. Должна получиться следующая структура папок проекта:

```
practicum_mui_pages
|——— node_modules
|——— public
|——— src
|      |—— components
|      |     |——— Navbar.tsx
|      |     |——— Footer.tsx
|      |
|      |—— images
|      |     |——— image1.jpg
|      |     |——— ...
|      |     |——— image10.jpg
|      |
|      |—— list
|      |     |——— components
|      |     |
|      |     |——— List.tsx /* страница со списком зданий */
|      |     |——— table.tsx /* скопировать из архива к ЛР*/
|      |
|      |——— main
|      |      |——— components
|      |      |      |——— Gallery.tsx
|      |      |      |——— Content.tsx
|      |      |      |——— BuildCard.tsx
|      |      |
|      |      |——— Main.tsx /* главная страница */
|      |
|      |——— App.tsx
|      |——— data.tsx
|      |——— index.tsx
```

2. Создать компонент **Main** в файле **main.tsx**:

```
import Navbar from "../components/Navbar";
import Gallery from "./components/Gallery";
import Content from "./components/Content";

function Main() {
  return (
    <div>
      <Navbar active="1"/>
      <Gallery/>
      <Content/>
    </div>
  );
}

export default Main;
```

3. Внести изменения в компонент **Content** (папка **main/components**): изменить путь в импорте файла с данными:

```
import structures from "../../data";
```

4. Внести изменения в компонент **Gallery** (папка **main/components**):: изменить путь в импорте файла с данными:

```
import structures from "../../data";
```

5. Внести изменения в файл **App.tsx** (вызвать главную страницу):

```
import Main from "./main/Main";

function App() {
  return (
    <>
      <Main/>
    </>
  );
}

export default App;
```

6. Перейти в папку проекта **practicum_mui_pages**, импортировать **react-router-dom**, библиотеки для визуализации данных и запустить приложение:

```
npm install react-router-dom
npm install @mui/x-data-grid
npm install @mui/x-charts
npm start
```

В результате в браузере по адресу **http://localhost:3000/** должна отобразиться главная страница приложения **Самые высокие здания и сооружения** (как в предыдущей Лабораторной работе).

Задание 2. Разработка страницы Список зданий

Создать страницу **Список зданий**. На странице вывести навигационную панель и список зданий в виде таблицы, использовать **`DataGrid`** (рисунок 1).

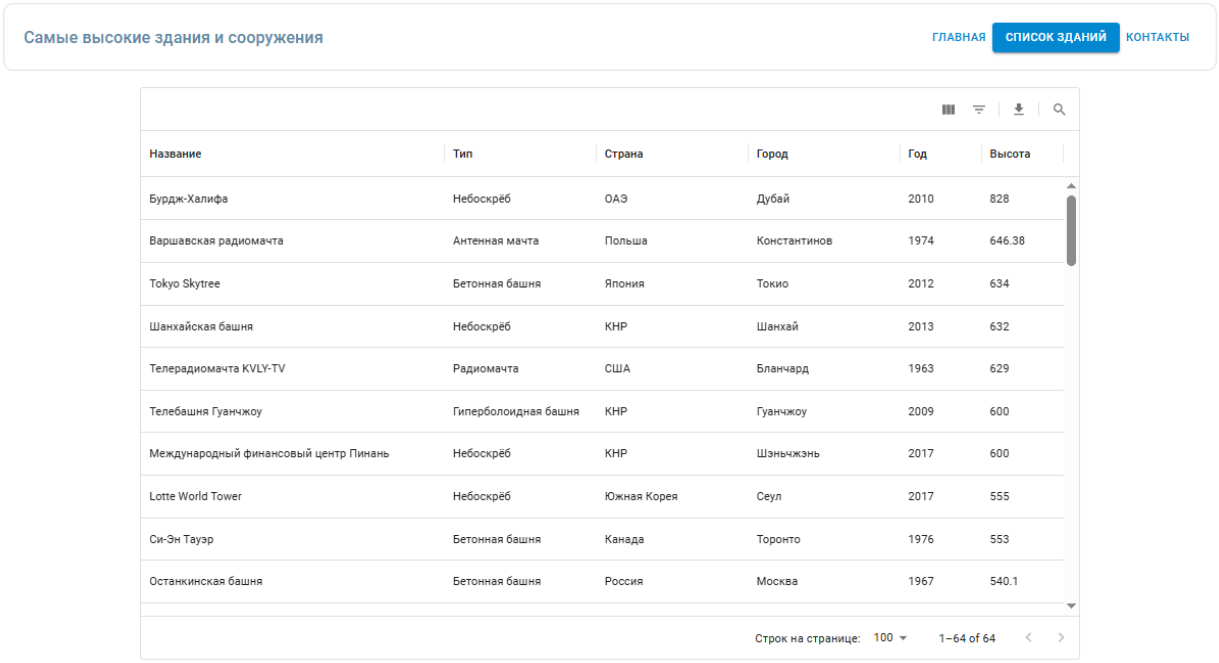


Рисунок 1. Данные с пагинацией, возможностью сортировки, фильтрации, экспорта

Порядок выполнения практического задания

1. Создать компонент **`List`** в файле **`list.tsx`**. Включить в него компонент **`Navbar`** с активной страницей **Список зданий**.

```
import Navbar from "../components/Navbar";

function List() {
  return (
    <div>
        <Navbar active="2"/>
    </div>
  );
}

export default List;
```

2. В файле **`App.tsx`** вызвать этот компонент, предварительно импортировав его:

```
import List from "./list/List";

function App() {
  return (
    <>
      <List/>
    </>
  );
}
```

В результате страница будет выглядеть, как показано на рисунке 2.

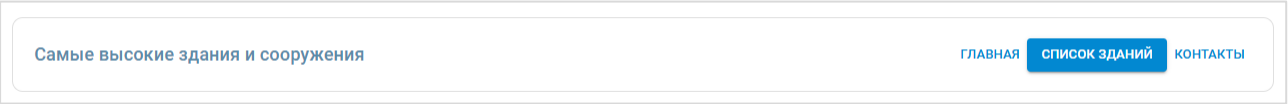


Рисунок 2. Навигационная панель на странице **Список зданий**

3. Создадим компонент **BuildingsGrid** в файле **BuildingsGrid** (папка **list/components**), используя компонент MUI **DataGrid**. В компоненте необходимо определить пропс **rows** (массив **buildings** из файла **table.tsx**) и пропс **columns** (ключи элемента массива **buildings**). При описании столбцов обязательно нужно указать, на основе каких ключей строится каждый столбец, также можно задать название столбца на странице (если ключ и название совпадают – название на странице можно не указывать).

```
import buildings from "../table";
import { DataGrid, GridRowsProp, GridColDef  } from "@mui/x-data-grid";

function BuildingsGrid() {

    const rows: GridRowsProp = buildings;

    const columns: GridColDef[] = [
        { field: 'Название', headerName: 'Название'},
        { field: 'Тип'},
        { field: 'Страна'},
        { field: 'Город',},
        { field: 'Год' },
        { field: 'Высота'},
    ];

    return (

          <DataGrid
            rows={rows}
            columns={columns}
          />

  );
}

 export default BuildingsGrid;
```

В компонент **List** добавим компонент **BuildingsGrid**.

```
import Navbar from "../components/Navbar";
import BuildingsGrid from "./components/BuildingsGrid";

function List() {
  return (
    <div>
        <Navbar active="2"/>
        <BuildingsGrid/>
    </div>
  );
}
```

В результате на странице отобразятся данные массива **buildings** в виде таблицы с пагинацией (рисунок 3).

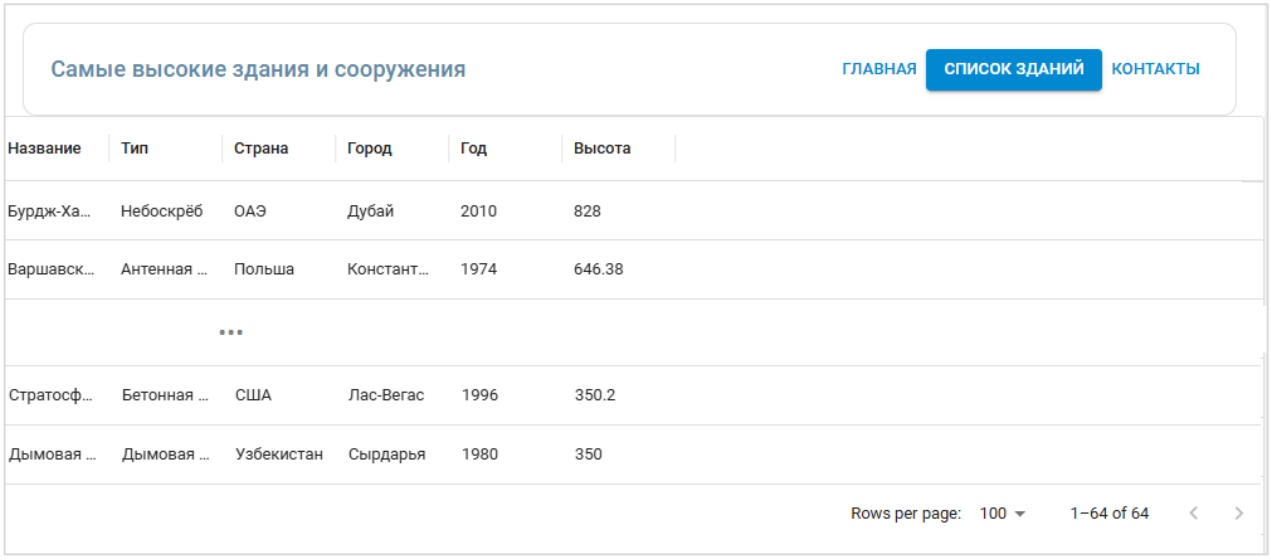


Рисунок 3.Таблица с данными

4. «Переведем» все подписи в компоненте **BuildingsGrid** на русский. Для этого укажем свойство **localeText** для **DataGrid:**

```
...
import { ruRU } from '@mui/x-data-grid/locales';

function BuildingsGrid() {
    ...
    return (
          <DataGrid
            localeText={ruRU.components.MuiDataGrid.defaultProps.localeText}
            rows={rows}
            columns={columns}
          />
  );
}
```

В результате все подписи в компоненте будут отображаться на русском языке.

5. Сделаем таблицу адаптивной, для этого заключим ее в компонент **Container** с максимальной шириной **lg**, зададим высоту контейнера и отступ от навигационной панели. Также для столбцов укажем адаптивную ширину.

```
...
import Container from '@mui/material/Container';

function BuildingsGrid() {

    const rows: GridRowsProp = buildings;

    const columns: GridColDef[] = [
        { field: 'Название', headerName: 'Название', flex: 1},
        { field: 'Тип', flex: 0.5},
        { field: 'Страна', flex: 0.5},
        { field: 'Город', flex: 0.5},
        { field: 'Год' },
        { field: 'Высота'},
    ];
```

```
    return (
        <Container maxWidth="lg" sx={{height: '700px', mt: '20px'}}>
          <DataGrid
            localeText={ruRU.components.MuiDataGrid.defaultProps.localeText}
            rows={rows}
            columns={columns}
          />
        </Container>
  );
```

В результате ширина таблицы при изменении ширины окна браузера будет адаптивно изменяться, таблица расположится по центру (рисунок 4).

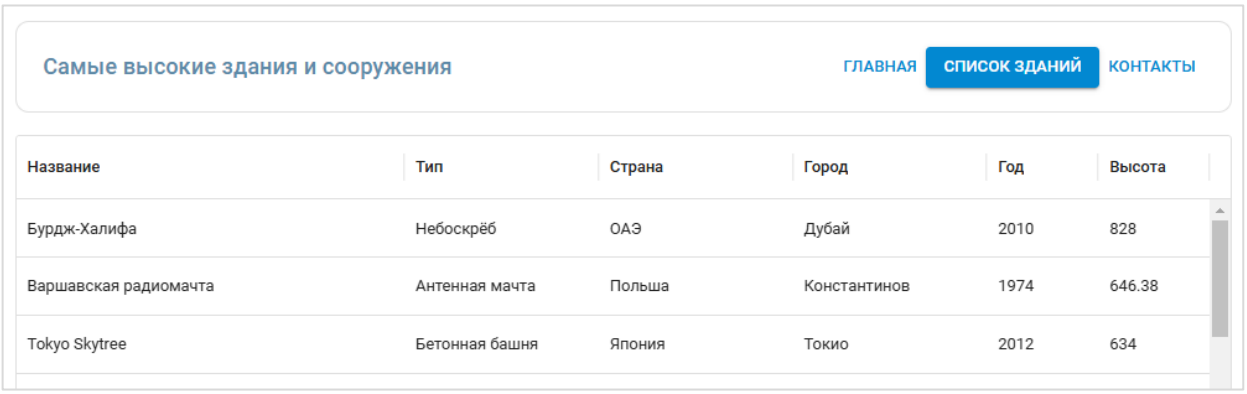


Рисунок 4. Адаптивная таблица с данными

6. Для всех столбцов таблицы реализованы операции сортировки, фильтрации др. Для вывода контекстного меню для каждого столбца необходимо кликнуть на символ ⋮ рядом с названием столбца (рисунок 5). Для выбранного столбца можно указать способ сортировки и фильтр.

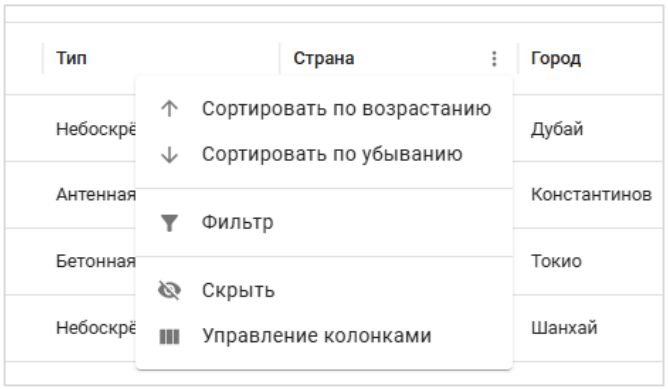
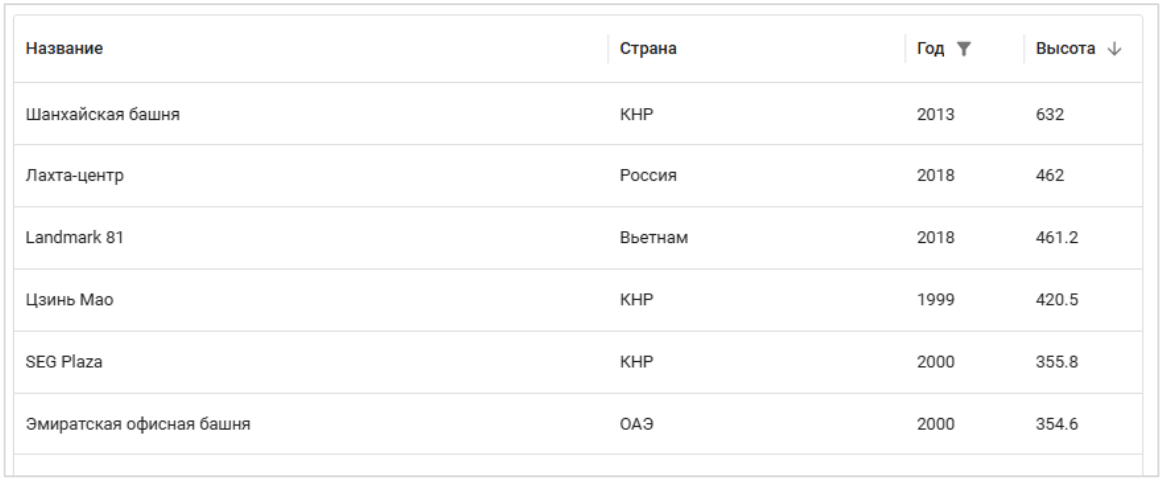


Рисунок 5. Контекстное меню для столбца Страна

Самостоятельное задание. С помощью контекстного меню получите следующие данные на странице (рисунок 6):



Название	Страна	Год	Высота
Шанхайская башня	КНР	2013	632
Лахта-центр	Россия	2018	462
Landmark 81	Вьетнам	2018	461.2
Цзинь Мао	КНР	1999	420.5
SEG Plaza	КНР	2000	355.8
Эмиратская офисная башня	ОАЭ	2000	354.6

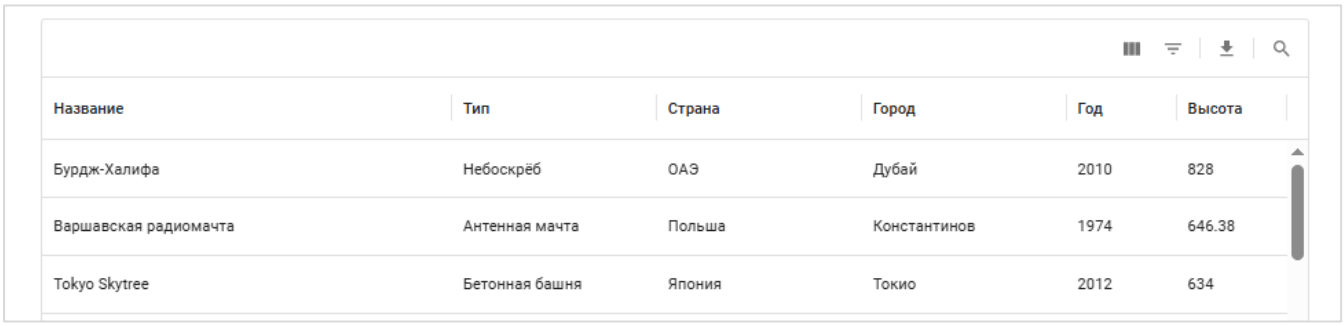
Рисунок 6. Отфильтрованные и отсортированные данные

7. Над таблицей добавим панель с инструментами (**Столбцы**, **Фильтры**, **Экспорт**). Для этого в компоненте **`BuildingsGrid`**: установим для ее отображения **`true`**:

```
function BuildingsGrid() {
    ...

    return (
        <Container maxWidth="lg" sx={{height: '700px', mt: '20px'}}>
          <DataGrid
            ...
            showToolbar={true}
            ...
          />
        </Container>
  );
}
```

В результате панель с инструментами будет размещена перед таблицей (рисунок 7).

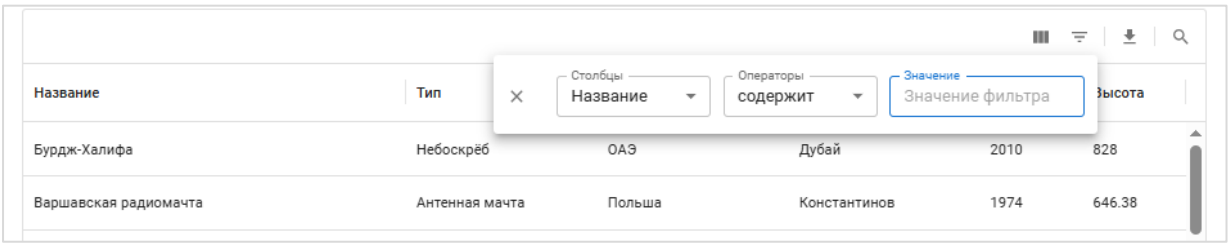


Название	Тип	Страна	Город	Год	Высота
Бурдж-Халифа	Небоскрёб	ОАЭ	Дубай	2010	828
Варшавская радиомачта	Антенная мачта	Польша	Константинов	1974	646.38
Tokyo Skytree	Бетонная башня	Япония	Токио	2012	634

Рисунок 7. Таблица с панелью с инструментами

С помощью инструментов этой панели можно фильтровать данные (рисунок 8а), выбирать отображаемые столбцы (рисунок 8б).

а)



б)

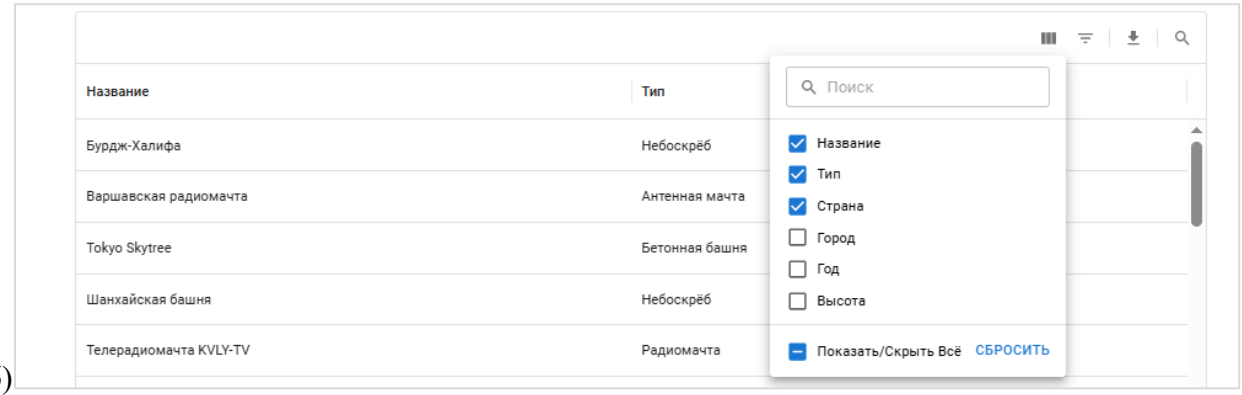


Рисунок 8. Таблица с панелью с инструментами

Самостоятельное задание. Отберите сооружения **Тип**, которых содержит слово «мачта», отсортируйте их по убыванию **Высоты**, скройте столбцы **Город**, **Год** и импортируйте эти данные в CSV формате. В загруженном файле **`React App.csv`** должны содержаться следующие данные:

```
Название,Тип,Страна,Высота
Варшавская радиомачта,Антенная мачта,Польша,646.38
Телерадиомачта KVLY-TV,Радиомачта,США,629
Киевская телебашня,Решётчатая мачта,Украина,385
Виннцкая телемачта,Радиомачта,Украина,354
```

Задание 3. Переходы между страницами

Реализовать возможность перехода по страницам при клике по соответствующий пункт меню.

Порядок выполнения практического задания

1. В файле **index.tsx** импортируйте **createBrowserRouter** и **RouterProvider** из **react-router-dom** для создания логики маршрутизатора.

```
...
import {
  createBrowserRouter,
  RouterProvider,
} from "react-router-dom";

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

2. Создайте объект-маршрутизатор, в котором для путей «**/**» и «**/list**» определите компоненты для соответствующих страниц. Импортируйте компоненты страниц.

```
...
import {
  createBrowserRouter,
  RouterProvider,
} from "react-router-dom";

import List from "./list/List";
import Main from "./main/Main";

const router = createBrowserRouter([
  {
    path: "",
    element: <Main />,
  },
  {
    path: "/list",
    element: <List />,
  },
]);

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);

root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

3. Добавьте **RouterProvider** в функцию рендеринга.

```
...
root.render(
  <React.StrictMode>
    <RouterProvider router={router} />
  </React.StrictMode>
);
```

4. В компонент **Navbar** (папка components) добавьте импорт компонента **Link**, оберните кнопки меню в **Link**. Укажите путь, в соответствии с которым будет выбран загружаемый компонент.

```
...
import {Link} from 'react-router-dom';

...
function Navbar({ active } : ComponentProps) {
    ...
    return (

        ...
                <Link to="/">
                  <Button … >
                    Главная
                  </Button>
                </Link>

                <Link to="/list">
                  <Button  …>
                    Список зданий
                  </Button>
                </Link>
        ...
    );
}
```

В результате при загрузке страницы должна отображаться **Главная** страница.

При клике на кнопки меню **Главная** и **Список зданий** должна выводиться соответствующая страница.

Самостоятельное задание

1. Исправьте компонент **Navbar**, чтобы при клике на элементы свернутого меню также отображались соответствующие страницы.

Самостоятельное задание 4

Создать страницу для отображения информации об одном здании на основе одного элемента массива **`structures`** (файл **`data.tsx`**). Эта страница должна выводиться по клику на картинку в галерее на **Главной** странице. Например, при клике на рисунок **Ворота запада, США** должна быть выведена страница, показанная на рисунке 9.

При клике на ссылку **`Главная`** (в навигационной цепочке в левой части страницы под навигационной панелью) необходимо перейти на **Главную** страницу. Также должны работать ссылки перехода на страницы **Главная** и **Список зданий** через меню.

При клике на другие рисунки должна отобразиться та же страница, но с другим содержанием.

Рисунок 9. Страница с описанием одного здания

Порядок выполнения самостоятельного практического задания

1. Добавьте в структуру папок проекта папку для страницы **`building`**. Для отображения страницы создайте компонент **`Building`**. Контент страницы построить на основе первого элемента массива **`structures`**. Реализовать адаптивность страницы при изменении ширины окна браузера.

2. В объекте-маршрутизаторе для пути «**`/building`**» укажите, что будет вызываться разработанный компонент **`Building`**.

3. Внесите изменения в компонент **`Gallery`**, чтобы при клике на любой рисунок открывалась страница с описанием первого элемента массива **`structures`**.

4. Добавьте возможность передавать параметр при вызове страницы (номер элемента массива **structures**):

- в роутере укажите параметр **id** в пути на страницу:

```
const router = createBrowserRouter([
  ...
  {
    path: "/building/:id",
    element: <Building />,
  },
]);
```

- в **Gallery** присоедините к пути индекс картинки в массиве из параметров **map**:

```
<Link key={ index } to={ "/building/" + index }>
```

- в **Building** получите параметр из строки с адресом, далее используйте его как номер элемента массива **structures**.

```
...
import { useParams, Link } from 'react-router-dom';

function Building() {

    const { id } = useParams();

  return (
    ...
  );
}
```

Задание 5

Создать страницу для отображения данных массива **buildings** в группированном виде. Группировка может осуществляться по столбцам **Страна**, **Год** или **Тип**. Для группированных данных вычислены значения **Максимальной**, **Минимальной** и **Средней высоты**. Массив сгруппированных данных (например, по **Стране**) и тип данных размещены в файле **groupdata.tsx** (прикреплен в архиве к Лабораторной работе):

```
export type tGroup = {
      "id": number,
      "Группа": string | number,
      "Минимальная высота": number ,
      "Максимальная высота": number ,
      "Средняя высота": number,
  }[];

export const countries: tGroup = [
    {
        "id": 1,
        "Страна": "Вьетнам",
        "Минимальная высота": 461,
        "Максимальная высота": 461,
        "Средняя высота": 461
    },
    {
        "id": 2,
        "Страна": "Германия",
        "Минимальная высота": 368,
        "Максимальная высота": 368,
        "Средняя высота": 368
    },
    ...
]
```

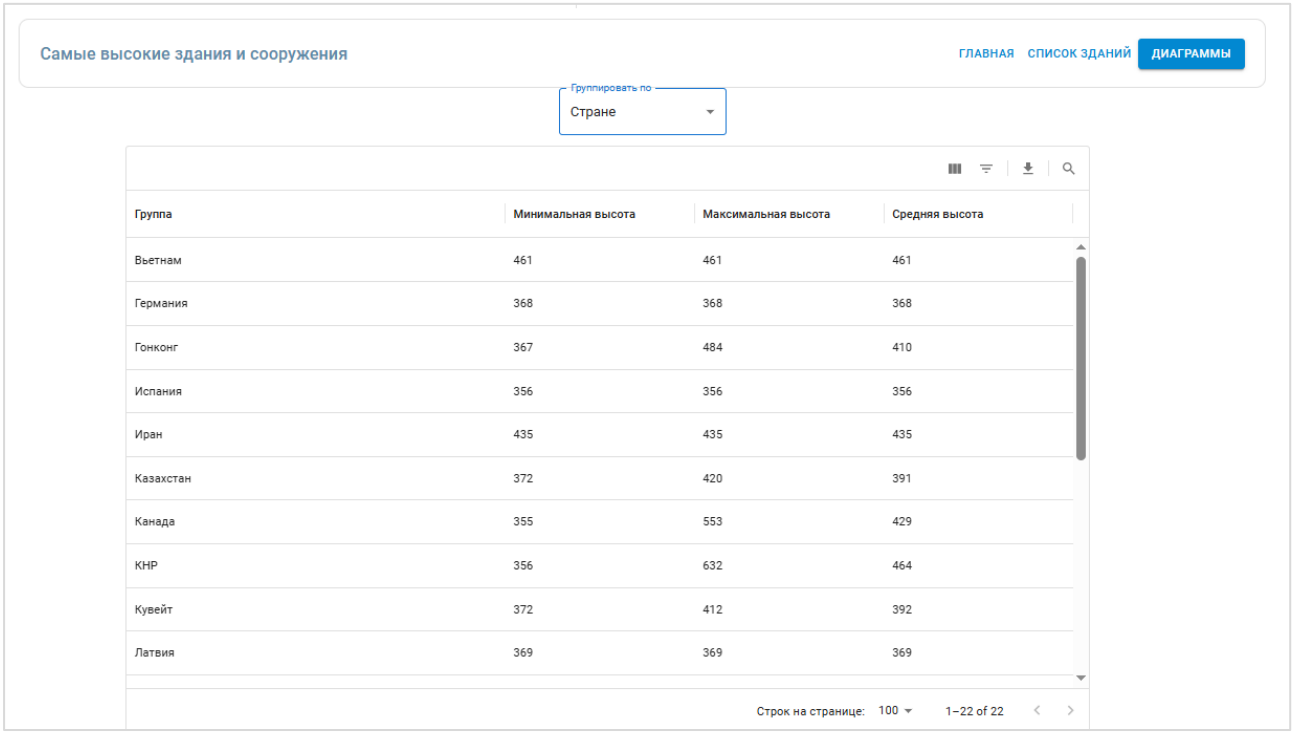


Рисунок 10. Страница с группированными данными

На страницу поместить поле со списком с перечнем столбцов, по которым группируются данные. После выбора пользователем нужного столбца на странице в виде **DataGrid** должны отображаться данные из соответствующего массива (рисунок 10).

Порядок выполнения практического задания

1. **Самостоятельно** добавить в проект страницу **Диаграммы**:

- в структуру проекта включить папку **chart**:

```
practicum_mui_pages
|——— node_modules
|——— public
|——— src
|      |—— components
...
|      |
|      |——— chart
|      |      |——— components
|      |      |      |——— GroupGrid.tsx /* компонент для вывода данных */
|      |      |
|      |      |——— Chart.tsx /* страница с диаграммой */
|      |      |——— groupdata.tsx /* сгруппированные данные */
|      |
...
|      |——— App.tsx
|      |——— data.tsx
|      |——— index.tsx
```

- создать компонент **Chart** для отображения страницы, в него включить компонент **Navbar** (в компоненте элемент **Контакты** заменить на **Диаграммы**), сделать активной страницу **Диаграммы**;

- реализовать возможность перехода к ней через меню с любой страницы проекта;

- отобразить на странице данные из массива **countries** в виде **DataGrid**. Для этого создать компонент **GroupGrid**, который в качестве пропса **data** получает массив **countries**, включить его в компонент **Chart** (*порядок создания компонента описан в Задании 2 данной лабораторной работы*).

Пояснение. Чтобы описать тип пропса компонента **GroupGrid**, воспользуйтесь типом **tGroup**, который определен в файле **groupdata.tsx**:

```
...
import { tGroup }  from "../groupdata";

type GroupProps = {
  data: tGroup;
};

function GroupGrid({ data }: GroupProps) {
  ...
}
```

В результате страница должна выглядеть, как показано на рисунке 10 (без поля со списком).

2. На страницу включить поле со списком для выбора столбца группировки данных. На страницу поместим компонент **Box**, в нем определим компонент **FormControl**, который

будет включать метку поля со списком и компонент **Select**. Элементы списка оформим в виде **MenuItem**:. При загрузке страницы в поле со списком выведем элемент **Страна**.

```
import Select, { SelectChangeEvent } from '@mui/material/Select';
import Box from '@mui/material/Box';
import InputLabel from '@mui/material/InputLabel';
import MenuItem from '@mui/material/MenuItem';
import FormControl from '@mui/material/FormControl';

function Chart() {

  return (
      <div>
        ... Вставить навигационную панель ...
        <Box sx={{ width:"200px", m:"auto" }}>
          <FormControl fullWidth>
            <InputLabel> Группировать по </InputLabel>
            <Select
              id="select-group"
              value="Страна"
              label="Группировать по"
            >
              <MenuItem value="Страна"> Стране </MenuItem>
              <MenuItem value="Год"> Году </MenuItem>
              <MenuItem value="Тип"> Типу </MenuItem>
            </Select>
         </FormControl>
        </Box>
        ... Вставить футтер ...
    </div>
  );
}
```

В результате на странице отобразиться поле со списком со значением **Страна**, но при выборе других элементов (**Тип**, **Год**) на странице ничего происходить не будет.

3. Добавим в компонент состояние **group** и функцию для его изменения. В качестве состояния может выступать одно из трех значений (**Страна**, **Год**, **Тип**). Объявим перечислимый тип для описания состояния. В начальный момент времени состоянию присвоим значение **Страна**. Свойству **value** компонента **Select** присвоим это состояние.

```
...
import * as React from 'react';

type tSelect = "Страна" | "Год" | "Тип";

function Chart() {
  const [group, setGroup] = React.useState<tSelect>("Страна");

  return (
      <div>
        ...
            <Select
              id="select-group"
              value={ group }
              label="Группировать по"
            >
             ...
            </Select>
        ...
    </div>
```

```
  );
}
```

4. При изменении поля со списком будем менять состояние **group**.

```
...
function Chart() {
  const [group, setGroup] = React.useState<tSelect>('Страна');
  const handleChange = (event: SelectChangeEvent) => {
    setGroup(event.target.value as tSelect);
  }

  return (
      <div>
        ...
            <Select
              id="select-group"
              value={ group }
              label="Группировать по"
              onChange={ handleChange }
            >
             ...
            </Select>
        ...
    </div>
  );
}
```

В результате поле со списком будет изменяться, но таблица с группированными данными меняться не будет.

5. Добавить в компонент состояние **groupData**, в котором будет храниться один из массивов с группированными данными (**years, countries, types**), и функцию его изменения. **Самостоятельно** включить изменение состояния **groupData**. В зависимости от выбранного элемента поля со списком в состояние занести соответствующий массив. Также передать состояние **groupData** в качестве пропса в компонент **GroupGrid**.

```
...
import {years, countries, types }  from "./groupdata";
...
function Chart() {
  const [group, setGroup] = React.useState<tSelect>('Страна');

  const [groupData, setGroupData] = React.useState(countries);

  const handleChange = (event: SelectChangeEvent) => {
    setGroup(event.target.value as tSelect);

    /* самостоятельно включить изменение состояния groupData */

  };

  return (
    <div>
      ...
      <GroupGrid data={groupData} />
    </div>
  );
}
```

В результате при изменении поля со списком в таблицу с данными будет выводиться соответствующий массив.

Задание 6

На страницу включить столбчатую диаграмму (гистограмму), визуализирующую сгруппированные данные по выбранному в поле со списком столбцу (рисунок 11).

Добавить элементы управления, с помощью которых можно указать, какие значения (**Максимальная**, **Минимальная** и/или **Средняя высота**) показывать на диаграмме.

Рисунок 11. Диаграммы на странице

Порядок выполнения практического задания

1. Создать компонент **`GroupChart`**. Вызвать его в компоненте **`Chart`** перед компонентом для вывода таблицы. Компонент **`GroupChart`** должен в качестве пропса **`data`** получать массив группированных данных, сохраненный в состоянии **`groupData`** компонента **`Chart`**.

2. В компонент **`GroupChart`** включить контейнер, максимальной ширины **`lg`**. В контейнер поместить компонент **`BarChart`** для построения столбчатой диаграммы. Диаграмма будет строиться по массиву **`data`** (свойство **`dataset`**), по оси OX будут отложены значения столбца, по которому осуществлялась группировка (свойство **`xAxis`**). На графике в виде столбиков будут отложены значения максимальной, минимальной и средней высоты зданий для каждого сгруппированного значения (свойство **`series`**). Также для области вывода диаграммы необходимо указать высоту и подпись оси OY **Высота(м)**, «отодвинув» ее на 10 пикселей влево от оси (**`chartSetting`**).

```
import { BarChart} from '@mui/x-charts/BarChart';
import Container from '@mui/material/Container';
...
```

```
function GroupChart({ data }: ...) {

  const chartSetting = {
    yAxis: [{ label: 'Высота (м)' }],
    height: 400,
  };

      };

    return(
        <Container maxWidth="lg">
          <BarChart
            dataset={ data }
            xAxis={[{ scaleType: 'band', dataKey: 'Группа' }]}
            series={[
                { dataKey: 'Минимальная высота', label: 'Минимальная высота'},
                { dataKey: 'Средняя высота', label: 'Средняя высота'},
                { dataKey: 'Максимальная высота', label: 'Максимальная высота'},
            ]}
            {...chartSetting}
            />
        </Container>
    )
}

export default GroupChart;
```

В результате на странице будет выведена диаграмма, на которой отображены значения максимальной, минимальной и средней высоте зданий для каждого сгруппированного значения. На рисунке 12 по оси OX отложены страны.

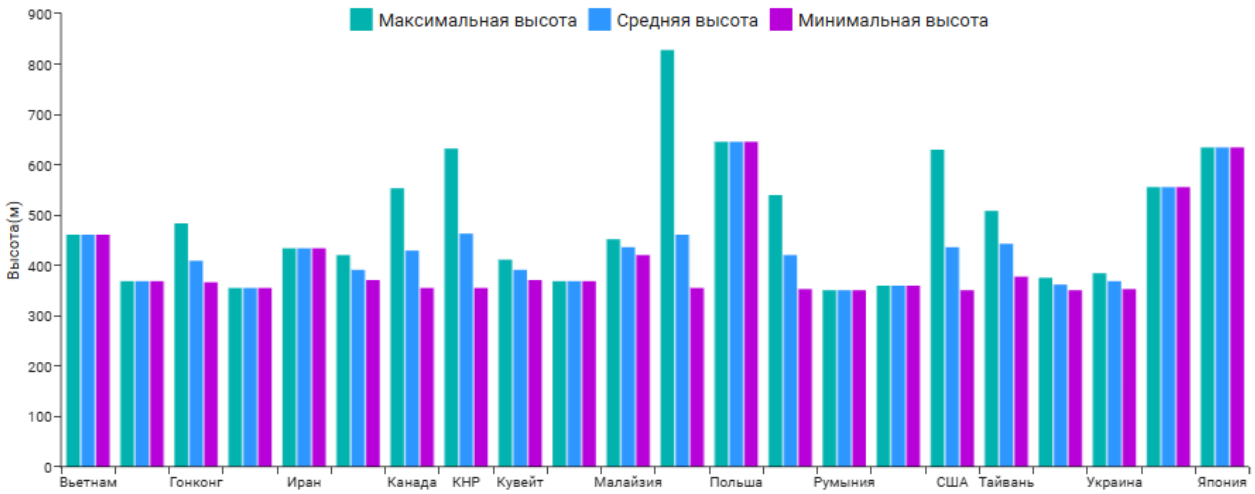


Рисунок 12. Столбчатая диаграмма

3. Расположим легенду рядов данных под диаграммой (свойство **slotProps**).

```
function GroupChart({ data }: ...) {
    ...
    return(
        <Container maxWidth="lg">
          <BarChart
            dataset={ data }
            xAxis={[{ scaleType: 'band', dataKey: 'Группа' }]}
```

```
            series={[
                { dataKey: 'Минимальная высота', label: 'Минимальная высота' },
                { dataKey: 'Средняя высота', label: 'Средняя высота' },
                { dataKey: 'Максимальная высота', label: 'Максимальная высота'
},
            ]}
            slotProps={{
              legend: {
                position: { vertical: 'bottom', horizontal: 'center' },
              },
            }}
            {...chartSetting}
            />
        </Container>
    )
}
```

В результате легенда расположиться под диаграммой, как показано на рисунке 11.

4. Чтобы управлять выводом значений на графике, в компонент **GroupChart** включим состояние **series**, где каждому ряду данных будут присвоено значение **true**, если ряд на диаграмму выводить надо и **false**, в противном случае. Определим функцию для управления состоянием **setSeries**.

```
...
function GroupChart({ data }: ...) {

    const [series, setSeries] = React.useState({
      'Максимальная высота': true,
      'Средняя высота': false,
      'Минимальная высота': false,
    });
    ...
    return(
      ...
    )
}
```

5. Сформируем значение свойства **series** диаграммы в зависимости от состояния **series**. Для этого преобразуем объект состояния в массив, каждый элемент которого представляет собой ключ и значение, отфильтруем полученный массив так, чтобы в нем остались только элементы, значения которых равно **true**, а затем сформируем описание каждого ряда данных. Полученный массив укажем как значение свойства **series**.

```
function GroupChart({ data }: ...) {
    const [series, setSeries] = React.useState({
      'Максимальная высота': true,
      'Средняя высота': false,
      'Минимальная высота': false,
    });

    let seriesY = Object.entries(series)
        .filter(item => item[1] == true)
        .map(item => {
            return {"dataKey": item[0], "label": item[0]}
        });
    ...
    return(
        <Container maxWidth="lg">
          <BarChart
```

```
            dataset={ data }
            xAxis={[{ scaleType: 'band', dataKey: 'Группа' }]}
            series={ seriesY }
            ...
            {...chartSetting}
            />
        </Container>
    )
}
```

В результате, если менять значение состояния, должны меняться выводимые ряды данных на диаграмме. При загрузке страницы будет выведен ряд данных **Максимальная высота** для выбранного в поле со списком столбца.

6. Для управления выводом рядов данных на диаграмму создадим компонент **SettingChart**. Этот компонент в качестве пропсов будет получать состояние series и функцию для его изменения. Компонент будет включать **FormControl** - обязательный компонент для ввода данных через форму. В нем можно указать подпись формы и включить различные элементы управления. Для управления выводом каждого ряда данных будем использовать **CheckBox**, обернутый в **FormControlLabel**. Тот элемент, который в состоянии отмечен как **true**, будет отмечен в свойстве **checked**.

```
import FormControl from '@mui/material/FormControl';
import FormLabel from '@mui/material/FormLabel';
import FormControlLabel from '@mui/material/FormControlLabel';
import Checkbox from '@mui/material/Checkbox';

type tSeries= {
        'Максимальная высота': boolean,
        'Средняя высота': boolean,
        'Минимальная высота': boolean,
    }

type CheckboxProps = {
    series: tSeries;
    setSeries: React.Dispatch<
      React.SetStateAction<tSeries>
    >;
};

function SettingChart({series, setSeries}: CheckboxProps) {

    return (
        <FormControl>
          <FormLabel id="label-checkbox-group">
            На диаграмме показать:
          </FormLabel>
          <FormControlLabel
            control={
              <Checkbox checked={series["Максимальная высота"]}
               name="Максимальная высота" />
            }
            label="максимальную высоту" />
          <FormControlLabel
            control={
              <Checkbox checked={series["Средняя высота"]}
               name="Средняя высота" />
            }
            label="среднюю высоту" />
          <FormControlLabel
```

```
            control={
              <Checkbox checked={series["Минимальная высота"]}
               name="Минимальная высота" />
            }
            label="минимальную высоту" />
        </FormControl>
    )
}

export default SettingChart;
```

В компоненте **GroupChart** вызвать **SettingChart** с необходимыми пропсами (предварительно импортировав его):

```
<SettingChart series={ series } setSeries={ setSeries }/>
```

В результате на странице отобразится группа **checkbox** (рисунок 13). Первый элемент будет отмечен. Но выбрать другие переключатели или снять первый будет невозможно.

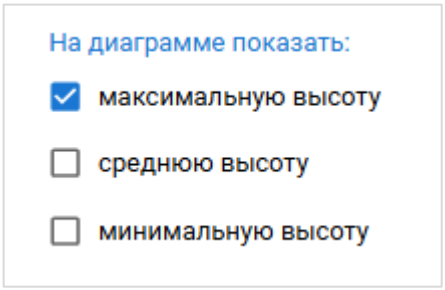


Рисунок 13. Группа **checkbox**

7. Обработаем событие **change** на каждом **checkbox**. Для этого в функции **handleChange** изменим состояние **series** в соответствии с выбором пользователя.

```
function SettingChart({series, setSeries}: CheckboxProps) {

    const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
        setSeries({
          ...series,
          [event.target.name]: event.target.checked,
        });
      };

    return (
       ...
          <FormControlLabel
            control={
              <Checkbox checked={series["Максимальная высота"]}
               onChange={handleChange} name="Максимальная высота" />
            }
            label="максимальную высоту" />
          <FormControlLabel
            control={
              <Checkbox checked={series["Средняя высота"]}
               onChange={handleChange} name="Средняя высота" />
            }
            label="среднюю высоту" />
          <FormControlLabel
```

```
            control={
              <Checkbox checked={series["Минимальная высота"]}
                onChange={handleChange} name="Минимальная высота" />
            }
            label="минимальную высоту" />
      ...
    )
}
```

В результате при клике на **checkbox** соответствующий ряд будет показываться/скрываться на диаграмме (рисунок 11).

Самостоятельное задание

1. В том случае, если на диаграмме отображается один любой ряд данных (рисунок 14), вывести подписи данных (свойство **barLabel="value"** компонента **BarChart**).

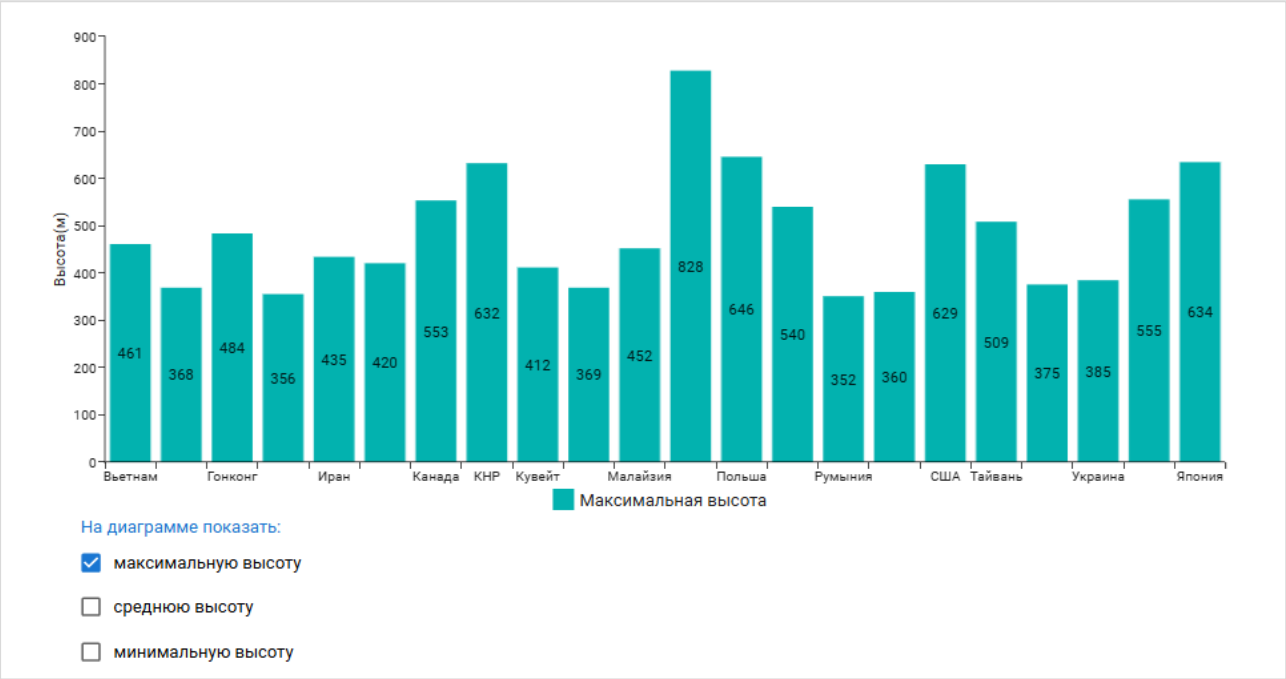


Рисунок 14. Диаграмма с подписями данных

Задание 7

Добавить на страницу **Диаграммы** возможность выбора типа диаграммы (**Гистограмма** или **Линейная**). При выборе пользователем типа диаграммы Линейная, страница должна выглядеть, как показано на рисунке 15.

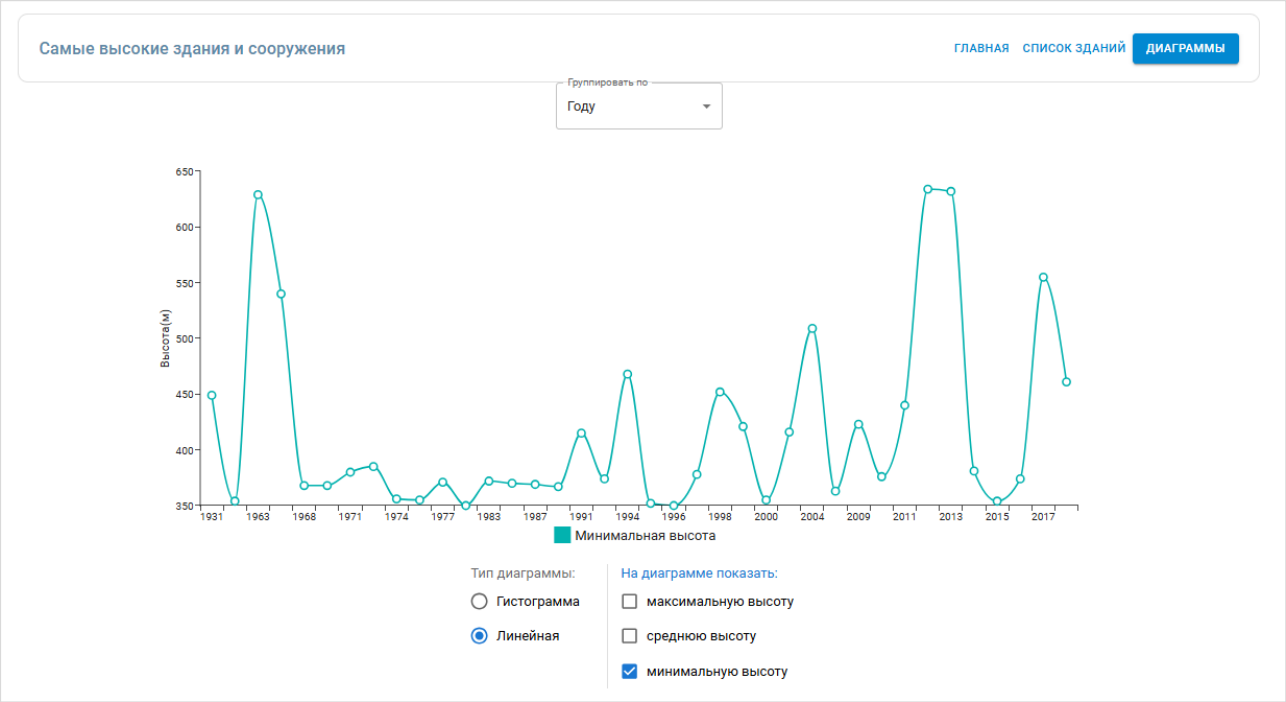


Рисунок 15. Визуализация данных с помощью линейной диаграммы

Порядок выполнения практического здания

1. В компонент **`GroupChart`** добавить построение линейной диаграммы.

```
...
import { LineChart} from '@mui/x-charts/LineChart';

function GroupChart({ data }: ...) {
      return(
      <Container maxWidth="lg">
        ...
          <LineChart
            dataset={ data }
            xAxis={[{ scaleType: 'band', dataKey: 'Группа' }]}
            series={ seriesY}
            slotProps={{
              legend: {
                position: { vertical: 'bottom', horizontal: 'center' },
              },
            }}
            {...chartSetting}
            />
        ...
      </Container>
    )
}
```

В результате на странице отобразятся 2 диаграммы (рисунок 16).

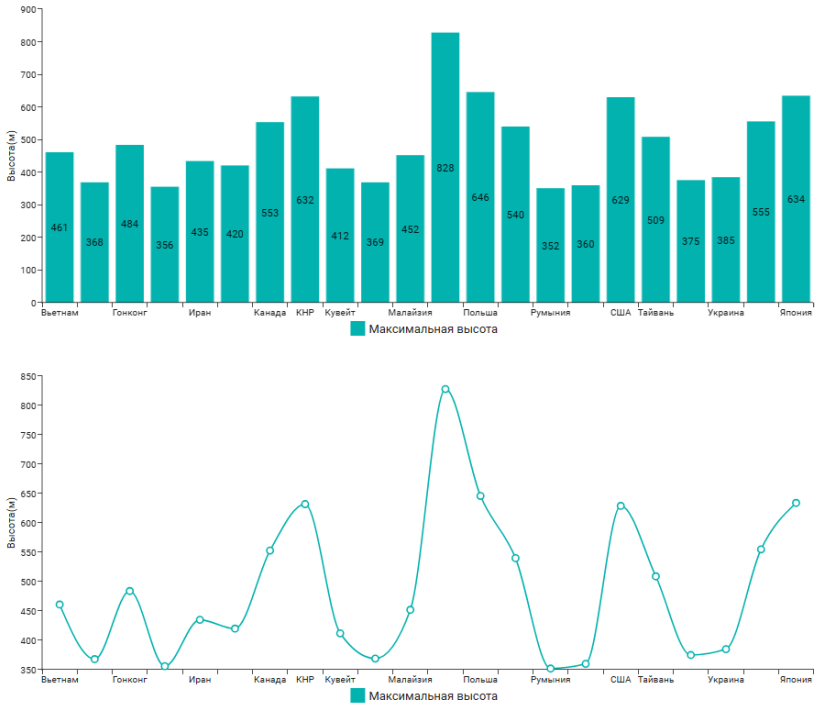


Рисунок 16. Столбчатая и линейная диаграммы

2. На странице должна быть выведена одна из диаграмм. Для управления выводом добавим в компонент **GroupChart** состояние **isBar**, которое будет иметь значение **true**, если на странице должна показываться столбчатая диаграмма, и **false**, если линейная. При загрузке страницы будет отображаться столбчатая диаграмма.

```
function GroupChart({ data }: ...) {
    ...
    const [isBar, setIsBar] = React.useState(true);
    ...

    return(
       ...
    )
}
```

3. **Самостоятельно** исправить код компонента **GroupChart** так, чтобы на страницу выводилась одна диаграмма в зависимости от значения состояния **isBar**.

4. В компонент **SettingChart** добавить группу кнопок **radio** для управления выводом диаграммы. В качестве пропосов в компонент передать состояние **isBar** и функцию для его изменения. Разместить группу перед группой с **checkbox**. Обернуть обе группы в компонент **Stack**, в котором задать выравнивание дочерних компонент, разделитель и др. свойства.

```
...
import Stack from '@mui/material/Stack';
import Divider from '@mui/material/Divider';
import RadioGroup from '@mui/material/RadioGroup';
import Radio from '@mui/material/Radio';

type CheckboxProps = {
```

```
    ...
    isBar: boolean;
    setIsBar: React.Dispatch<
      React.SetStateAction<boolean>
    >;
};

function SettingChart({series, setSeries, isBar, setIsBar}: CheckboxProps) {

   ...
    return (
      <Stack
        direction="row"
        justifyContent="center"
        divider={<Divider orientation="vertical" flexItem />}
        spacing={2}
        sx={{  m: "20px 0" }}
      >
        <FormControl>
          <FormLabel id="label-radio-group">
            Тип диаграммы:
          </FormLabel>
          <RadioGroup
            name="group-radio"
            value={(isBar) ? "bar": "dot"}
          >
            <FormControlLabel value="bar"
              control={
                <Radio checked={isBar} />
              }
              label="Гистограмма"
            />
            <FormControlLabel value="dot"
              control={
                <Radio checked={!isBar}/>
              }
              label="Линейная" />
          </RadioGroup>
        </FormControl>

        <FormControl>
           ...
        </FormControl>

      </Stack>
    )
}
```

В результате на страницу выводится диаграмма в зависимости от состояния **isBar**. Переключить кнопки в группе пока нельзя.

Самостоятельное задание

1. Обработать событие **change** на каждом **radio**. Изменить состояние **isBar** при клике на переключатель. В результате на страницу должна выводиться диаграмма в зависимости от выбора пользователя.