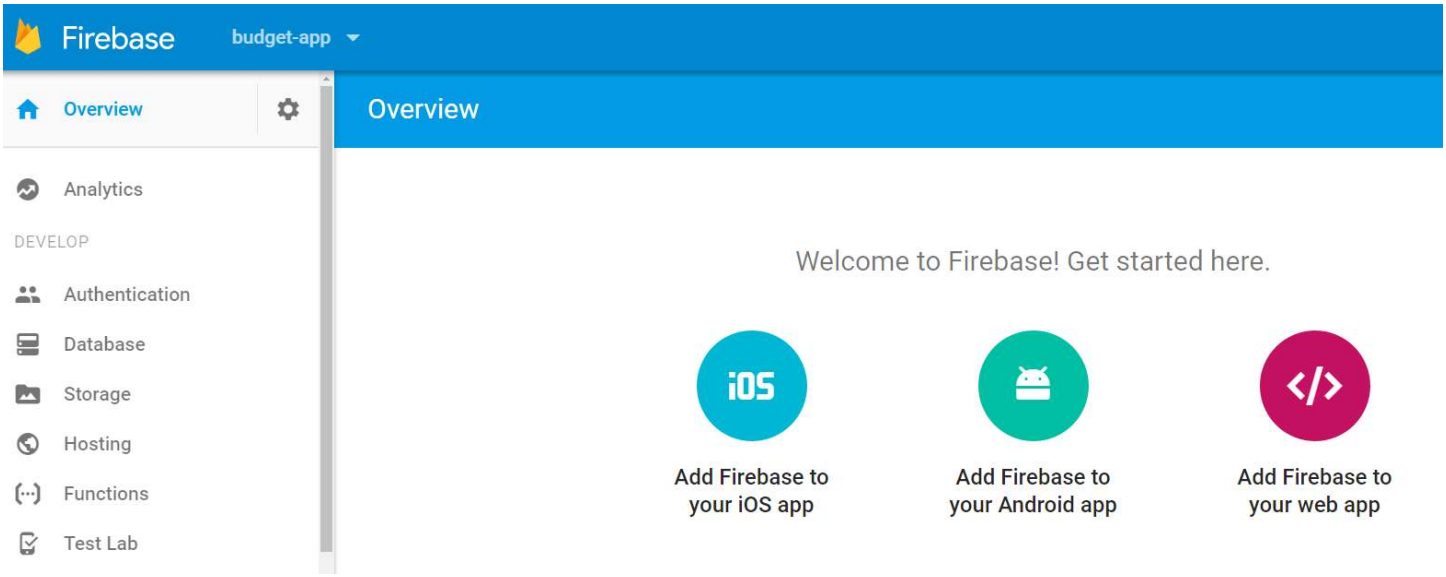
STEP 1: Start Project, add dependencies and config database

create new project change to project directory install dependencies	ng new angular-part2 cd angular-part2 npm install --save firebase primeng font-awesome
Add PrimeNG Stylesheets	edit src/angular-cli.json and add the following to the styles section (~line 23) "../node_modules/primeng/resources/primeng.min.css", "../node_modules/primeng/resources/themes/omega/theme.css", "../node_modules/font-awesome/css/font-awesome.min.css"

Add Firebase

Firebase Console Overview Tab click the Add Firebase to your web app.

Click the copy button to copy the code to the clipboard



Import the firebase	**Edit src/app/app.component.ts and add the following import (~line 2)** import * as firebase from "firebase";

Add database connection
 	
Note: Replace the config with your own on the clipboard
	Edit src/app/app.component.ts and add the following constructor (~line 12)
	constructor(){
	let config = {
	apiKey: "AIzaSyAr3Bg2tJBrf_c9o6W0EK1B17RiHbu1hPw",
	authDomain: "budget-app-7f40c.firebaseapp.com",
	databaseURL: "https://budget-app-7f40c.firebaseio.com",
	storageBucket: "budget-app-7f40c.appspot.com",
	messagingSenderId: "792611408752"
	};
	}

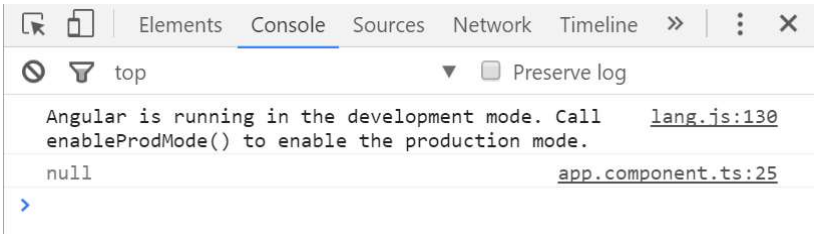
```
  firebase.initializeApp(config);

  let root = firebase.database().ref();
  root.on('value', function (snap){
    console.log(snap.val())
  });
}
```

Verify Firebase

ng serve

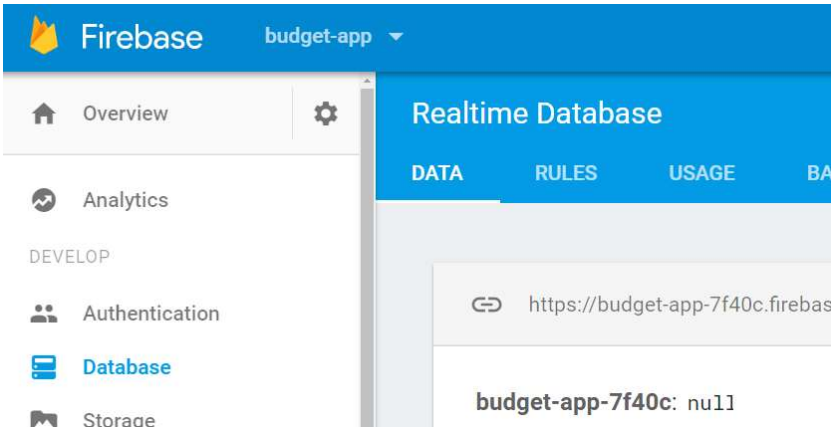
in browser goto localhost:4200 and open the developer tools console

you should see the value null on the console.

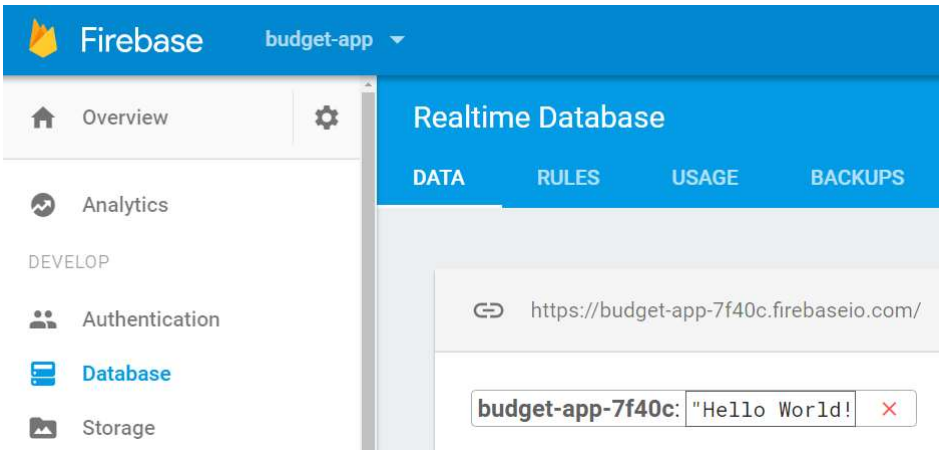


In the firebase console click database on the side menu

You will notice the value of the database in **null**

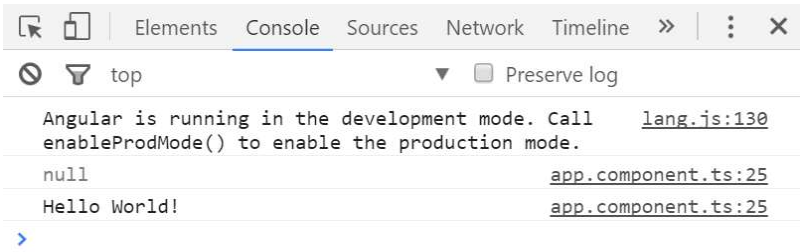


Click the null value and change it to **Hello World!**



Switch back to the app in the browser:

The application has been updated with the database change.



STEP -2 Initialize Some Real Data for the Application

The application is a budget application which needs a list of budget categories.

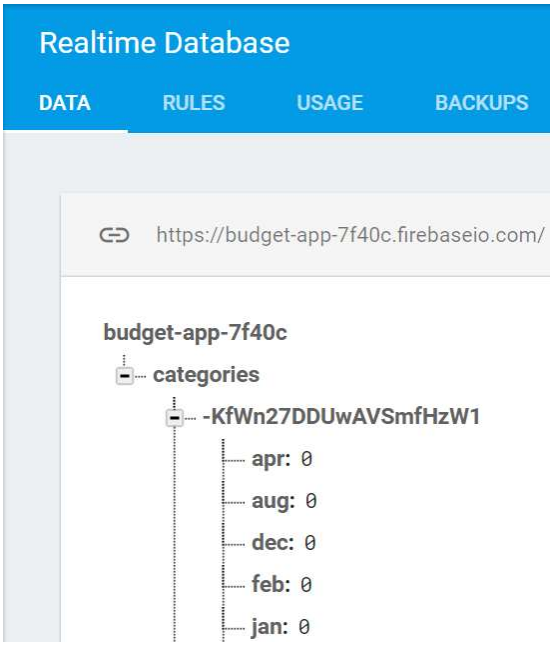
	Edit src/app/app.component.ts and add initDB method (~line 29)

initDB function will initialize database with categories	initDB() {
	let categories = firebase.database().ref('categories');
	let monthlyBudget = firebase.database().ref('monthly-budget');
	
	let budgetCategories = ['Mortgage/Rent', 'Electricity', 'Mobile Phone', 'Cable',
	'Groceries', 'Entertainment', 'Water/Sewer', 'Auto Loan', 'Dining Out',
	'Auto Ins', 'HO Ins', 'Rainy Day Fund', 'Vacation Fund', 'Retirement']
	let months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
	'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
	budgetCategories.forEach(category => {
	const categoryRef = categories.push({name: category});
	months.forEach(month => {
	monthlyBudget.push({month: month, amount: 0, category: categoryRef.key})
	})

	`})` `}`
`Button to initialize db`	**Edit src/app/app.component.html add a button (~line 4)** `<button (click)="initDB()">Initialize DB</button>`

Return to the browser and click the InitDB Button

Switch to firebase database view and the data will be listed



Step 3: Create App UI

Import PrimeNG Components to be used

`Add PrimeNG Imports`	**Edit src/app/app.module.ts add imports at (~line 5)**
	`import {InputTextModule} from 'primeng/primeng';`
	`import {ButtonModule} from 'primeng/primeng';`
	`import {ChartModule} from 'primeng/primeng';`
	`import {DataTableModule, SharedModule} from 'primeng/primeng';`

`Import PrimeNG modules`	**Edit src/app/app.module.ts add imports at (~line 20)**
	`,`
	`InputTextModule,`
	`ButtonModule,`
	`ChartModule,`
	`DataTableModule,`
	`SharedModule`

NOTE: stop and start ng-serve as these changes are not automatically reloaded

Button using PrimeNG Style
 	
PrimeNG dataTable
 	
Columns to be displayed with numbers editable
	Edit src/app/app.component.html change the existing InitDB Button to PrimeNG and add DataTable
	`<button pButton type="button" (click)="initDB()" label="Initialize DB"></button>`
	`<p-dataTable [value]="budgetData" [editable]="true" (onEditComplete)="updateData($event) ">`
	`<p-column field="name" header="Category" ></p-column>`
	`<p-column field="jan" header="January" [editable]="true" ></p-column>`
	`<p-column field="feb" header="February" [editable]="true"></p-column>`
	`<p-column field="mar" header="March" [editable]="true"></p-column>`
	`<p-column field="apr" header="April" [editable]="true"></p-column>`
	`<p-column field="may" header="May" [editable]="true"></p-column>`
	`<p-column field="jun" header="June" [editable]="true"></p-column>`
	`<p-column field="jul" header="July" [editable]="true"></p-column>`
	`<p-column field="aug" header="August" [editable]="true"></p-column>`
	`<p-column field="sep" header="September" [editable]="true"></p-column>`
	`<p-column field="oct" header="October" [editable]="true"></p-column>`
	`<p-column field="nov" header="November" [editable]="true"></p-column>`
	`<p-column field="dec" header="December" [editable]="true" ></p-column>`
	`</p-dataTable>`

Test DataGrid

Change the value of one of the values and check the database.

Change a value in database and check the App DataTable.

Step 4: Generate Chart

PrimeNG Chart is dependant on ChartJS, Add the dependency and chartjs script to the project	`npm install --save chart.js`
	edit /src/angular-cli.json and add the following to scripts (~line 27)
	`"../node_modules/chart.js/dist/Chart.js"`

Add chart to html	Edit src/app/app.component.html (~line 4)
	`<p-chart #chart type="pie" [data]="chartData" width="200" height="200" [options]="chartOptions"></p-chart>`
	Edit p-datatable to update chart (~line 7)
	`<p-dataTable [value]="budgetData" [editable]="true" (onEditComplete)="updateData($event, chart)" (onRowClick)="updateChart($event, chart)">`

Code for chart	src/app/app.component.ts
	properties (~line 12)
	`chartData: any;`
	`chartOptions: any;`
	`selectedRow: any;`
	constructor (~line 16)

```
  this.chartOptions = {
    responsive: false,
    maintainAspectRatio: false
  }
```

Update Chart at end of updateData method (~line 77)

```
  this.updateChart(event, chart)
```

add updateChart method (~line 80)

```
updateChart(event, chart) {
  if (this.selectedRow !== event.data) {
    this.selectedRow = event.data;

    let labels = ['jan', 'feb', 'mar', 'apr', 'may', 'june',
      'jul', 'aug', 'sep', 'oct', 'nov', 'dec'
    ];

    let data = {
      labels: labels,
      datasets: [
        {
          data: [event.data.jan,
            event.data.feb,
            event.data.mar,
            event.data.apr,
            event.data.may,
            event.data.jun,
            event.data.jul,
            event.data.aug,
            event.data.sep,
            event.data.oct,
            event.data.nov,
            event.data.dec
          ],
          backgroundColor: [
            "#E60012",
            "#F39800",
            "#FFF100",
            "#8FC31F",
            "#009944",
            "#009E96",
            "#00A0E9",
```

```
            "#0068B7",
            "#1D2088",
            "#920783",
            "#E4007F",
            "#E5004F"
          ],
          hoverBackgroundColor: [
            "#E60012",
            "#F39800",
            "#FFF100",
            "#8FC31F",
            "#009944",
            "#009E96",
            "#00A0E9",
            "#0068B7",
            "#1D2088",
            "#920783",
            "#E4007F",
            "#E5004F"
          ]
        }]
    };

    this.chartData = Object.assign({}, data);
    setTimeout(() => {
      chart.reinit();
    }, 100);
  }
```