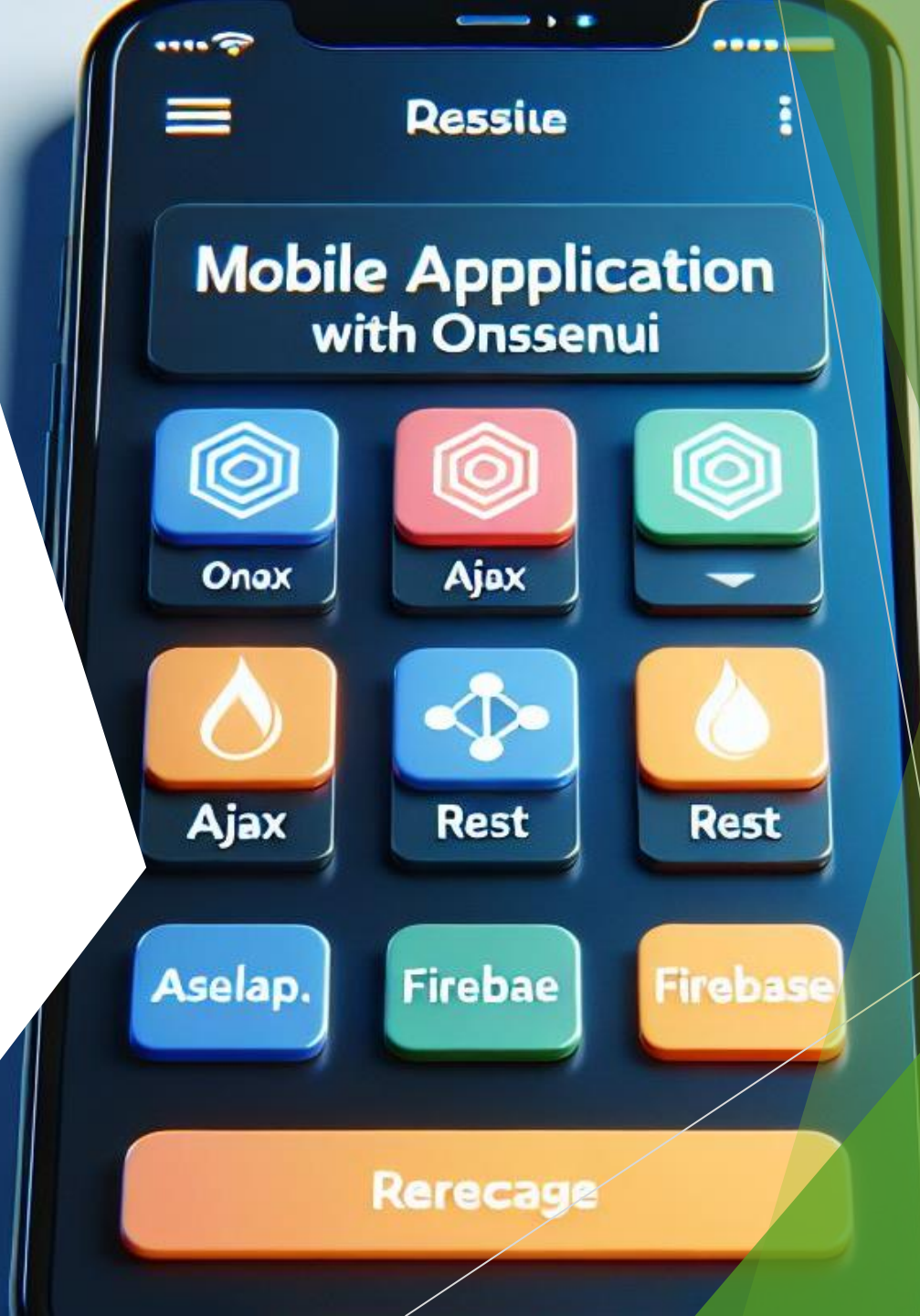


Създаване на  
приложение с  
OnsenUI, AJAX,  
REST и Firebase



# Разгледайте различните функционалности и възможности на Firebase

Firebase е популярна и използвана платформа за разработка на уеб и мобилни приложения, създадена и управлявана от Google. Тя носи множество предимства за разработчиците, благодарение на своите разнообразни функционалности. Нека разгледаме кои са те.

## Функционалности на Firebase

Основните функции на Firebase се разделят на три категории - *Build*, *Release & Monitor* и *Engage*. Всяка една от тези функционалности се използва в различен етап от разработката на приложения.

### Built features

Build включва широк набор от компоненти, които позволяват на разработчиците да покрият всички изисквания при създаването на приложения. Тук са включени две различни бази данни, както и допълнителни функции за съхранение на данни, автентикация и изкуствен интелект.



Firebase

Firebase дава достъп до две бази данни:  
**Cloud Firestore**, наричана още Google Firestore е cloud-базирана NoSQL база, за съхранение и синхронизация на данни.  
**Realtime Database** е cloud-базирана база данни, която улеснява съхранението на JSON-базирани данни изпълнява синхронизация в реално време. Части от Realtime Database функционират като клиенти по време на процесите за разработка на приложения, използващи iOS, JavaScript и Android SDK.

**Изкуствен интелект (Machine learning)**

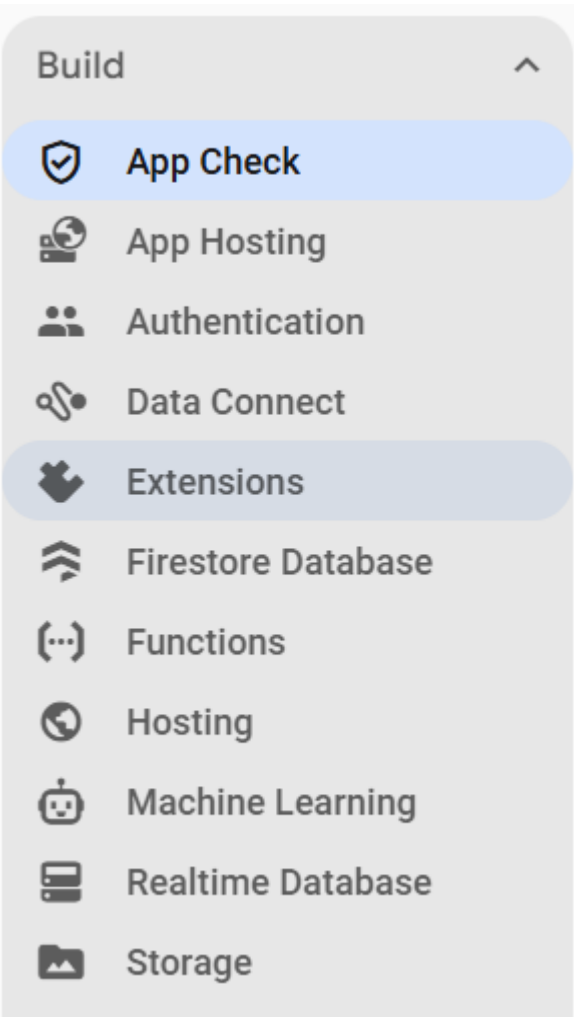
Firebase Machine learning представлява мобилен SDK, който използва силата на машинното обучение на Google и го прилага към приложенията за iOS и Android.  
Използването на тази функционалност с Firebase е подходящо за всеки разработчик, независимо от опита, който има.

**Cloud - функции**

Firebase cloud функциите представляват serverless framework, който позволява на разработчиците да приложат бекенд кода, за да отговаря на HTTPS и Firebase ивенти.  
Google cloud се използва за съхранение на Typescript и JavaScript код, който може да бъде изпълнен в управляваната среда.

**Автентикация**

Автентикацията е още една функция на Firebase, която предлага готови за използване UI библиотеки, бекенд и SDK за автентикация на потребителите в приложението. Това се случва чрез телефонен номер, парола или други приложения като Google, Twitter, Facebook и т.н.



## Съобщения

Firebase cloud messaging или FCM представлява услуга за изпращане на безплатни съобщения. Собствениците на апликацията могат да изпращат нотификации на своите потребители, за да привлекат вниманието им.

## Хостинг

Firebase предоставя няколко възможности за хостинг за Microservices, уеб апликации и други видове съдържание. Потребителите получават достъп до SSL защита, а съдържанието може да бъде хоствано в различни категории.

## Съхранение

Cloud storage е функция на Firebase, която позволява съхранение на различни ресурси, свързани с разработката на приложения. Потребителите могат напълно безопасно да качват и теглят различни видове файлове и съдържание.

## Емулатор

За улесняване на интеграцията и тестването на всички функции, Firebase предоставя и достъп до Local Emulator Suite, който позволява на разработчиците да изпробват кода. Emulator Suite предоставя емулятори за автентикация, cloud функции, Firestore, RTDB, хостинг и др.

Build





Run




 A/B Testing

 AdMob


 App Distribution


 Crashlytics

 Dynamic Links

 Messaging

 Performance

 Release Monitoring

 Remote Config

 Test Lab

Analytics





# The "Book Library" App

- ▶ Проектирайте и създайте приложение за библиотека в OnsenUI с REST back-end в Firebase
  - ▶ Книгите трябва да имат заглавие, автор и описание (title, author and description)
- ▶ Внедрете следните функционалности :
  - ▶ Вход, регистрация, изход, всички книги, създаване нова книга, редактирайте съществуваща, изтриване
- ▶ Всеки може да разгледа всички книги
- ▶ Само съзателят на книгата може да редактира / изтрива собствените си книги



**Please login**

Username:

Password:

Login

**Please register here**

Username:

Password:

Register

Вход / Регистрация

# Екран на списъка с книги

[Home](#)[List Books](#)[Create Book](#)[Logout](#)

Welcome, guest!

Books loaded.


## Books

Title	Author	Description	Actions
Java Fundamentals	Bay Ivan	Strong Java book. Not for beginners	
Software Technologies	Bay Ivan	Creating simple apps with HTML5, PHP, C# and Java.	
C# Basics	Bay Ivan	Very good C# book!	
Some title  	Bay Ivan	Some text	<a href="#">[Delete]</a> <a href="#">[Edit]</a>
Some title  	Some author  	Some description  	<a href="#">[Delete]</a> <a href="#">[Edit]</a>

# Екран за създаване на книга

---

[Home](#) [List Books](#) [Create Book](#) [Logout](#) Welcome, guest!

**Create new book** 

Title:

Author:

Description:

New book description  
comes here...



# Екран за редактиране на книга

Home   List Books   Create Book   Logout  
Welcome, guest!

## Books

Title	Author	Description	Actions
Some title  	Some author  	Very good C# book!	
Some title  	Some author  	Some description  	<a href="#">[Delete]</a> <a href="#">[Edit]</a>



Home   List Books   Create Book   Logout   Welcome, guest!

## Edit existing book

Title:


Author:

Description:  

Some description <br>

Edit

# Realtime Database


 Need help with Realtime Database? Ask Gemini

Data

Rules

Backups

Usage


 Extensions



Protect your Realtime Database resources from abuse, such as billing fraud or phishing

[Configure App Check](#)



 `https://test-project-b78c0-default-rtdb.europe-west1.firebaseio.com`



`https://test-project-b78c0-default-rtdb.europe-west1.firebaseio.com`

▼ users  
 ▶ — -OJETM1PuhGf0hI\_IBdt  
 ▶ — -OJETVnp2ykgzpMDNrHP

# Firebase Realtime Database

# Създай нова книга

**POST** <https://<project>.firebase.database.app/books.json>

?auth=<Database secret>

```
{ "title": "ttt", "author": "aaa", "description": "ddd" }
```

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `https://test-project-b78c0-default-rtdb.europe-west1.firebaseio.com/books.json?auth=8kpIne3FnUNPfuVzu0AxVDnfeI7SDVxostYjcXDm`
- Body:** `{ "title": "t2", "author": "a2", "description": "d2" }`
- Response:** 200 OK, 235 ms, 323 B. The response body is a JSON object: `{ "name": "-OJEXAmTq1969w7wMKWj" }`.

# СПИСЪК НА ВСИЧКИ КНИГИ

**GET** <https://<projec>.firebase.database.app/books.json>  
**?auth=<Database secret>**

The screenshot shows a REST client interface. At the top, a GET request is defined with the URL `https://test-project-b78c0-default-rtdb.europe-west1.firebaseio.com/books.json?auth=8kplne3FnUNPfUVzu0AxVDnfeI7SDVxostYjcXDm`. The 'Body' tab is selected, showing a message: 'This request does not have a body'. Below the request, the response is displayed in the 'Body' tab, showing a JSON object with a single key-value pair. The status bar indicates a 200 OK response with a 62 ms latency and 364 B of data.

GET ⌵ `https://test-project-b78c0-default-rtdb.europe-west1.firebaseio.com/books.json?auth=8kplne3FnUNPfUVzu0AxVDnfeI7SDVxostYjcXDm` Send ⌵

Params ● Authorization Headers (6) Body Scripts Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

---

Body Cookies Headers (8) Test Results 200 OK • 62 ms • 364 B • ⋮

{} JSON ⌵ ▶ Preview 🔗 Visualize ⌵ ⌵ 🔍 🔗

```
1 {
2   "-OJEXAmTq1969w7wMKWj": {
3     "author": "a2",
4     "description": "d2",
5     "title": "t2"
6   }
7 }
```

PUT

https://<projec>.firebase.database.app/books.json

?auth=<Database secret>

{ "title": "t2", "author": "a2", "description": "d2" }

PUT ▼ https://test-project-b78c0-default-rtdb.europe-west1.firebaseio.com/books.json?auth=8kplne3FnUNPfuVzu0AxVDnfe17SDVxostYjcXDm Send ▼

Params ● Authorization Headers (8) Body ● Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Beautify

```
1 { "title": "ttt", "author": "aaa", "description": "ddd" }
```

Body Cookies Headers (8) Test Results 200 OK • 73 ms • 342 B • 🌐 ⋮

{} JSON ▼ ▶ Preview 🔗 Visualize ▼ 🔧 📄 🔍 🔗

```
1 {
2   "author": "aaa",
3   "description": "ddd",
4   "title": "ttt"
5 }
```



**DELETE**

<https://<projec>.firebaseDATABASE.app/books.json>

```
{ "title":"t2", "author":"a2", "description":"d2" }
```

DELETE ▼ <https://test-project-b78c0-default-rtdb.europe-west1.firebaseio.com/books.json?auth=8kplne3FnUNPfuVzu0AxVDnfeI7SDVxostYjcXDm> Send ▼

Params ● Authorization Headers (8) Body ● Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Beautify

```
1 { "title":"ttt", "author":"aaa", "description":"ddd" }
```

Body Cookies Headers (8) Test Results 200 OK • 69 ms • 295 B • ⋮

{} JSON ▼ ▶ Preview 🔗 Visualize ▼

```
1 null
```

[Home](#)[Login](#)[Register](#)[List Books](#)[Create Book](#)[Logout](#)

Loading ...

Info

Error

## Welcome

Welcome to our book library.

# Създайте скелета на приложението

HTML, CSS, Views, Forms, Info Boxes

# Изглед за книгите (добавете OnsenUI)

```
<section id="viewBooks">
  <h1>Books</h1>
  <div id="books">
    <table>
      <tr>
        <th>Title</th>
        <th>Author</th>
        <th>Description</th>
        <th>Actions</th>
      </tr>
```

```
      <tr>
        <td>Book title</td>
        <td>Book author</td>
        <td>Book description</td>
        <td>
          <a href="#">[Delete]</a>
          <a href="#">[Edit]</a>
        </td>
      </tr>
      ...
    </table>
  </div>
</section>
```

## Books

Title	Author	Description	Actions
Book title	Book author	Book description	<a href="#">[Delete]</a> <a href="#">[Edit]</a>
Book title	Book author	Book description	<a href="#">[Delete]</a> <a href="#">[Edit]</a>

# Изглед за създаване на книга (добавете OnsenUI)

```
<section id="viewCreateBook">
  <h1>Create new book</h1>
  <form id="formCreateBook">
    <div>Title:</div>
    <div><input type="text" name="title" required /></div>
    <div>Author:</div>
    <div><input type="text" name="author" required /></div>
    <div>Description:</div>
    <div><textarea name="descr"
      rows="10" required></textarea></div>
    <div><input type="submit" value="Create" /></div>
  </form>
</section>
```

## Create new book

Title:

Author:

Description:

Create

# Изглед за редактиране на книга (добавете OnsenUI)

```
<section id="viewEditBook">
  <h1>Edit existing book</h1>
  <form id="formEditBook">
    <div><input type="hidden" name="id" required /></div>
    <div>Title:</div>
    <div><input type="text" name="title" required /></div>
    <div>Author:</div>
    <div><input type="text" name="author" required /></div>
    <div>Description:</div>
    <div><textarea name="descr" rows="10"
      required></textarea></div>
    <div><input type="submit" value="Edit" /></div>
  </form>
</section>
```

## Edit existing book

Title:

Author:

Description:

Edit





# Прилагане на CRUD операции

List / Create / Delete / Edit

# Списък на книгите: Заявка AJAX

```
function listBooks() {  
    document.getElementById('books').innerHTML = '';  
    showView('viewBooks');  
  
    fetch(Url)  
        .then(response => response.json())  
        .then(loadBooksSuccess)  
        .catch(handleAjaxError);  
}
```

```
function loadBooksSuccess(books) {
    showInfo('Books loaded. ');
    if (books.length === 0) {
        document.getElementById('books').textContent = 'No books in the
library.';
    } else {
        let booksTable = document.createElement('table');
        booksTable.classList.add('table', 'table-striped');

        let thead = document.createElement('tr');
        thead.innerHTML =
'<th>Title</th><th>Author</th><th>Description</th><th>Actions</th>';
        booksTable.appendChild(thead);

        books.forEach(book => {
            appendBookRow(book, booksTable);
        });

        document.getElementById('books').appendChild(booksTable);
    }
}
```

# Създайте нова книга: Заявка AJAX

```
export function createBook() {  
  let bookData = {  
    title: document.querySelector('#formCreateBook  
input[name=title]').value,  
    author: document.querySelector('#formCreateBook  
input[name=author]').value,  
    description: document.querySelector('#formCreateBook  
textarea[name=descr]').value  
  };  
}
```

# Създаване на книга: Заявка AJAX

```
function createBook() {  
  fetch(baseUrl, {  
    method: "POST" })  
    .then(response => {  
      if (!response.ok) {  
        throw new Error('Network response was not ok');  
      }  
      return response.json();  
    })  
    .then(createBookSuccess)  
    .catch(handleAjaxError);  
  
  function createBookSuccess(response) {  
    listBooks();  
    showInfo('Book created.');  }  
}
```



# Изтриване на книга: Заявка AJAX

```
function deleteBook(bookId) {  
  // Confirmation prompt  
  if (!window.confirm('Are you sure you want to delete this book?')) {  
    return; // If the user clicks 'Cancel', exit the function  
  }  
  
  fetch(baseUrl, {  
    method: "DELETE" })  
    .then(response => {  
      if (!response.ok) {  
        throw new Error('Network response was not ok');  
      }  
      return response.json();  
    })  
    .then(deleteBookSuccess)  
    .catch(handleAjaxError);  
  
  function deleteBookSuccess(response) {  
    listBooks();  
    showInfo('Book deleted.');  }  
}
```