UNIT 3 REST API

What is REST API (RESTful API)?

- An **API** is an *application programming interface*. It is a set of rules that allow programs to talk to each other. The developer creates the API on the server and allows the client to talk to it.
- REST determines how the API looks like. It stands for *"Representational State Transfer"*. It is a set of rules that developers follow when they create their API. One of these rules states that you should be able to get a piece of data (called a resource) when you link to a specific URL.
- **REST technology** is generally preferred over other similar technologies. This tends to be the case because REST uses less *bandwidth*, making it more suitable for efficient internet usage.

How does RESTful API work?

A **RESTful API** uses commands to obtain resources. The state of a resource at any given timestamp is called a resource representation. A RESTful API uses existing HTTP methodologies defined by the RFC 2616 protocol, such as:

- **GET** to retrieve a resource;
- **PUT** to change the state of or update a resource, which can be an object, file or block;
- **POST** to create that resource; and
- **DELETE** to remove it.

Architectural constraints of RESTful API

1. Uniform interface and underlaying methods of the network protocol.

2. Client - server based

3. Stateless Operations

4. RESTful resource caching

5. Layered system

6. Code on demand

1. Uses uniform interface (UI) and underlaying methods of the network protocol.

Resources should be uniquely identifiable through a single URL, and only by using the underlying methods of the network protocol, such as DELETE, PUT and GET with HTTP, should it be possible to manipulate a resource.

2. Client – server based

There should be a clear delineation between the client and server. UI and request-gathering concerns are the client's domain. Data access, workload management and security are the server's domain. This loose coupling of the client and server enables each to be developed and enhanced independent of the other.

3. Stateless Operations

All client-server operations should be stateless, and any state management that is required should take place on the client, not the server.

4. RESTful resource caching

All resources should allow caching unless explicitly indicated that caching is not possible.

5. Layered System

REST allows for an architecture composed of multiple layers of servers.

6. Code on demand

Most of the time, a server will send back static representations of resources in the form of XML or JSON. However, when necessary, servers can send executable code to the client.

Characteristics of REST API Architecture

1. Scalability

REST API offers great scalability. As the clients and servers are separated, the product might be scaled by the team of developers without much trouble.

Plus, it is also easier to integrate REST with present sites without refactoring website infrastructure. This allows developers to work faster instead of spending time reworking a website from scratch. As an alternative, they can merely add extra functionality.

2. Flexibility and Portability

Users can easily communicate even if the REST client server is hosted on different servers, offering an important benefit from the perspective of management.

3. Independence

The REST protocol makes it easy for developments across the different areas of a project to occur autonomously. Moreover, the REST API is adjustable to the operational syntax and platform, offering the prospect to test numerous environments during development.

UNIT 4 SOAP API

What is SOAP?

- ***Simple Object Access Protocol*** *or SOAP* is a standardized protocol that sends messages using other protocols such as HTTP, SMTP, TCP, or UDP.
- The SOAP specifications are official web standards, maintained and developed is an official protocol maintained by the *World Wide Web Consortium (W3C).*
- The main difference is that SOAP is a protocol while REST is not. Typically, an API will adhere to either REST or SOAP, depending on the use case and preferences of the developer.

How does SOAP API work?

When a request for data is sent to a SOAP API, it can be handled through any of the application layer protocols:

- **HTTP(for web browsers),**
- **SMTP (for email),**
- **TCP, and others.**

However, once a request is received, return SOAP messages must be returned as *XML documents* - a markup language that is both human- and machine-readable.

A completed request to a SOAP API is not cacheable by a browser, so it cannot be accessed later without resending to the API

SOAP API STRUCTURE

SOAP follows a formal and standardized approach that specifies how to encode XML files returned by the API. A SOAP message is, in fact, an ordinary XML file that consists of the following parts:

- ***Envelope (required)*** – This is the starting and ending tags of the message.
- ***Header (optional)*** – It contains the optional attributes of the message. It allows you to extend a SOAP message in a modular and decentralized way.
- ***Body (required)*** – It contains the XML data that the server transmits to the receiver.
- ***Fault (optional)*** – It carries information about errors occurring during processing the message.

```
<?xml version='1.0' Encoding='UTF-8' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
                 env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2007-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
                 env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
   <n:name>Fred Bloggs</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2007-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2007-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference></p:seatPreference>
   </p:return>
  </p:itinerary>
 </env:Body>
</env:Envelope>
```

SOAP Web specifications

Web services security (WS-security): Standardizes how messages are secured and transferred through unique identifiers called tokens.

WS-Reliable Messaging: Standardizes error handling between messages transferred across unreliable IT infrastructure.

Web services addressing (WS-addressing)*:* Packages routing information as metadata within SOAP headers, instead of maintaining such information deeper within the network.

Web services description language (WSDL): Describes what a web service does, and where that service begins and ends.

Characteristics of SOAP Architecture

SOAP has three major characteristics:

- ***Extensibility*** (security and WS-routing are among the extensions under development),
- ***Neutrality*** (SOAP can be used over any transport protocol such as HTTP, SMTP, TCP, or JMS), and
- ***Independence*** (SOAP allows for any programming model).

Since SOAP is a **protocol**, it imposes built-in rules that *increase its complexity and overhead*, it requires more bandwidth and resources which can lead to slower page load times.

However, these standards also offer built-in compliances that can make it preferable for enterprise scenarios.

The built-in compliance standards include ***security***, ***atomicity***, ***consistency***, ***isolation***, *and* ***durability*** *(****ACID****)*, which is a set of properties for ensuring reliable database transactions.

Difference between REST API and SOAP API

SOAP vs. REST

REST is a set of guidelines that offers flexible implementation, whereas SOAP is a protocol with specific requirements like XML messaging.

REST APIs are lightweight, making them ideal for newer contexts like the Internet of Things (IoT), mobile application development, and serverless computing.

SOAP web services offer built-in security and transaction compliance that align with many enterprise needs, but that also makes them heavier. Additionally, many public APIs, like the Google Maps API, follow the REST guidelines.

UNIT 5

Google Firebase

Firebase is an app development platform that helps you build and grow apps and games users love.

Google Firebase is Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

Google Firebase services - build

Build powerful apps. Spin up your backend without managing servers. Effortlessly scale to support millions of users with Firebase databases, machine learning infrastructure, hosting and storage solutions, and Cloud Functions.

Build Products – Cloud Firestore

Cloud Firestore - is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps - at global scale.

Query and structure data the way you like

Structure your data easily with collections and documents. Build hierarchies to store related data and easily retrieve the data you need using expressive queries. All queries scale with the size of your result set (note: not your data set), so your app is ready to scale from day one.

Buildtruly serverless apps

Cloud Firestore ships with mobile and web SDKs and a comprehensive set of security rules so you can access your database without needing to stand up your own server. Using Cloud Functions, our serverless compute product, you can execute hosted backend code that responds to data changes in your database. Of course, you can also access Cloud Firestore with traditional client libraries too (i.e. Node, Python, Go, and Java).

Sync data across devices, on or offline

With Cloud Firestore, you can automatically synchronize your app data between devices. We'll notify you of data changes as they occur so you can easily build collaborative experiences and realtime apps. Your users can access and make changes to their data at any time, even when they're offline. Offline mode is available on iOS, Android and Web!

Scale globally

Powered by Google's storage infrastructure, Cloud Firestore is built to scale with your business. Now, you can focus on building your app instead of managing servers or worrying about consistency.

Strong user-based security

With our declarative security language, you can restrict data access based on user identity data, pattern matching on your data, and more. Cloud Firestore also integrates with Firebase Authentication to give you simple and intuitive user authentication.

Build Products – Realtime database

- **Realtime Database** - Store and sync data with our NoSQL cloud database. Data is synced across all clients in real time and remains available when your app goes offline.
- **Real time syncing for JSON data**

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in realtime.

- **Collaborate across devices with ease**

Realtime syncing makes it easy for your users to access their data from any device: web or mobile, and it helps your users collaborate with one another.

- **Build serverless apps**

Realtime Database ships with mobile and web SDKs so you can build apps without the need of servers. You can also execute backend code that responds to events triggered by your database using Cloud Functions for Firebase.

- **Optimized for offline use**

When your users go offline, the Realtime Database SDKs use local cache on the device to serve and store changes. When the device comes online, the local data is automatically synchronized.

- **Strong user-based security**

The Realtime Database integrates with Firebase Authentication to provide simple and intuitive authentication for developers. You can use our declarative security model to allow access based on user identity or with pattern matching on your data.

Build Products – Remote config

Firebase Remote Config - is a cloud service that lets you change the behavior and appearance of your app without requiring users to download an app update. When using Remote Config, you create in-app default values that control the behavior and appearance of your app. Then, you can later use the Firebase console or the Remote Config backend APIs to override in-app default values for all app users or for segments of your user base. Your app controls when updates are applied, and it can frequently check for updates and apply them with a negligible impact on performance.

Make updates without republishing

Remote Config gives you visibility and fine-grained control over your app's behavior and appearance so you can make changes by simply updating its configuration from the Firebase console. This means you can dynamically turn features on and off, personalize by audience segments, and run experiments - all without setting up any complex infrastructure or releasing a new version.

Confidently roll out new features

Implement feature flags so you can gradually roll out new features to ensure they are stable and performing well, before rolling them out broadly. If the new features don't meet expectations or cause crashes, you can quickly roll them back. Remote Config provides signals and safeguards to ensure only the most engaging, high-quality features are released to all of your users.

Personalize your app for different audiences

Customize your app for different user segments based on their profile, preferences, past actions, and future predicted behavior. For example, you can tweak the homescreen for users based on their country. You can also simplify a game level for users who are predicted to churn to keep them engaged.

Run experiments to test ideas

Remote Config works seamlessly with A/B Testing so you can run experiments to test ideas and see the impact they have on key metrics. Want to see if your new checkout screen is actually driving more purchases? Want to try out a new ad format to see if it increases revenue? Use Remote Config with A/B Testing to get insight into which changes are helping you achieve your goals.

Build Products – authentication

Firebase Authentication - provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

Easy sign-in with any platform

Firebase Authentication aims to make building secure authentication systems easy, while improving the sign-in and onboarding experience for end users. It provides an end-to-end identity solution, supporting email and password accounts, phone auth, and Google, Twitter, Facebook, and GitHub login, and more.Run experiments to test ideas

Flexible, drop-in UI

FirebaseUI provides a customizable, open source, drop-in auth solution that handles the UI flows for signing in users. The FirebaseUI Auth component implements best practices for authentication on mobile devices and websites, which can maximize sign-in and sign-up conversion for your app.

Comprehensive security

Built by the same team that developed Google Sign-in, Smart Lock and Chrome Password Manager, Firebase security applies Google's internal expertise of managing one of the largest account databases in the world.

Fast implementation

It can take months to set up your own auth system, and it requires an engineering team to maintain that system in the future. Set up the entire authentication system of your app in under 10 lines of code, even handling complex cases like account merging.

Advanced features available with Google Cloud Identity Platform

We offer a seamless upgrade to Google Cloud Identity Platform for additional paid benefits and features like multi-factor authentication, blocking functions, and Enterprise SLAs.

Build Products – Cloud messaging

Firebase Cloud Messaging (FCM) - is a cross-platform messaging solution that lets you reliably send messages at no cost.

Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to drive user re-engagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4000 bytes to a client app.

Send messages to any device

Firebase Cloud Messaging (FCM) provides a reliable and battery-efficient connection between your server and devices that allows you to deliver and receive messages and notifications on iOS, Android, and the web at no cost.

Advanced message targeting

Easily target messages using predefined segments or create your own, using demographics and behavior. Target messages to devices that have subscribed to specific topics, or get as granular as a single device.

Customized notification content

Deliver notification messages immediately, or at a future time in the user's local time zone. Send custom data, and set priorities, sounds, and expiration dates, and track custom conversion events.

No coding required for sending notifications

Notification messages are fully integrated with Google Analytics for Firebase, giving you access to detailed engagement and conversion tracking. Monitor effectiveness from a single dashboard with no coding required.

A/B test notifications

Use A/B testing to try out different versions of your notification messages, and then easily see which one performs best against your goals. A/B testing can also be used with Remote Config.

A/B testing was built in partnership with Google Optimize, an A/B testing and personalization tool for the web.

Build Products – storage

Cloud Storage for Firebase - is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality.

Store your users' photos and videos

Cloud Storage is designed to help you quickly and easily store and serve user-generated content, such as photos and videos.

Build at Google scale

Our infrastructure is built for when your app goes viral. Effortlessly grow from prototype to production using the same technology that powers apps like Spotify and Google Photos.

Robust uploads and downloads

Your users aren't always online, so we built the Firebase SDK for Cloud Storage with mobile connectivity in mind. It will automatically pause and resume your transfers as the app loses and regains mobile connectivity, saving your users time and bandwidth.

Strong user-based security

The Firebase SDK for Cloud Storage integrates with Firebase Authentication to provide simple and intuitive access control. You can use our declarative security model to allow access based on user identity or properties of a file, such as name, size, content type, and other metadata.

Build Products – Extensions

Deploy functionality to your app quickly

Designed to increase productivity, Firebase Extensions provide extended functionality to your apps without the need to research, write, or debug code on your own.

Configure extensions to suit your use case

With Firebase Extensions, you provide the configuration parameters for your extension that are unique to your needs. You can also review the APIs enabled, resources created, and access granted to the extension.

Install easily

Extensions are open-sourced and built on Firebase and Google Cloud products you already know. Deployment and configuration of an extension are performed in the Firebase console or the Firebase CLI. Once deployed, they require no maintenance.

Build Products – app check

Protection for your data and your customers

App Check is an additional layer of security that helps protect access to your services by attesting that incoming traffic is coming from your app, and blocking traffic that doesn't have valid credentials. It helps protect your backend from abuse, such as billing fraud, phishing, app impersonation, and data poisoning.

Broad platform support that you can tailor to your needs

App Check supports Android, iOS and Web out of the box. For customers that want to support more platforms, you can integrate your own attestation provider with App Check's custom capabilities.

Integration with Firebase products, or your own backend

App Check works with Firebase products, like Cloud Firestore, Realtime Database, Cloud Storage for Firebase, Cloud Functions for Firebase, Google Cloud products like Cloud Run and Apigee, or your own custom API endpoints.

Fast, flexible and reliable

With a convenient UI and SDKs for web, iOS, and Android, you can get up and running with App Check with just two lines of code. App Check complements existing security solutions like Firebase Auth and Security Rules to provide even stronger security. And for developers with more sophisticated needs, App Check is certified under major compliance and security standards.

Build Products – cloud functions

Develop a backend without servers

Create functions that are triggered by Firebase products, such as changes to data in the Realtime Database, new user sign-ups via Auth, and conversion events in Analytics.

Run your mobile backend code without managing servers

Cloud Functions are single-purpose JavaScript functions that are executed in a secure, managed Node.js environment. They are only executed when a specific event being watched is emitted.

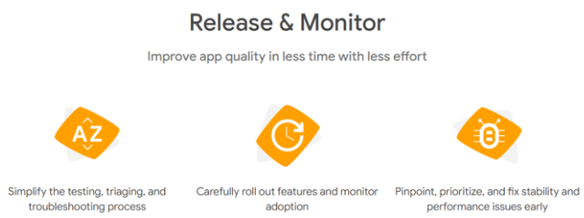
Low maintenance

Deploying your code to our servers requires just one command. After that, Cloud Functions automatically scales up computing resources to match the usage patterns of your app. You never worry about SSH credentials, server configuration, provisioning new servers, or decommissioning old ones.

Keeps your logic private and secure

In many cases, application logic is best controlled on the server to avoid tampering on the client side. Cloud Functions is fully insulated from the client, so you can be sure its functions are private and secure and can't be reverse engineered.

Release & monitor Products



Firebase Crashlytics - is a lightweight, realtime crash reporter that helps you track, prioritize, and fix stability issues that erode your app quality. Crashlytics saves you troubleshooting time by intelligently grouping crashes and highlighting the circumstances that lead up to them.

Firebase is Google Analytics - is an unlimited analytics solution available at no charge. Analytics integrates across Firebase features and provides you with unlimited reporting for up to 500 distinct events that you can define using the Firebase SDK.

Firebase Performance Monitoring - is a service that helps you to gain insight into the performance characteristics of your Apple, Android, and web apps.

Firebase App Distribution - makes distributing your apps to trusted testers painless. By getting your apps onto testers' devices quickly, you can get feedback early and often. And if you use Crashlytics in your apps, you'll automatically get stability metrics for all your builds, so you know when you're ready to ship.

Engage Products

Firebase A/B Testing - helps you optimize your app experience by making it easy to run, analyze, and scale product and marketing experiments. It gives you the power to test changes to your app's UI, features, or engagement campaigns to see if they actually move the needle on your key metrics (like revenue and retention) before you roll them out widely.

Dynamic Links - is for your users to get the best available experience for the platform they open your link on. If a user opens a Dynamic Link on iOS or Android, they can be taken directly to the linked content in your native app. If a user opens the same Dynamic Link in a desktop browser, they can be taken to the equivalent content on your website.

Firebase In-App Messaging - helps you engage your app's active users by sending them targeted, contextual messages that encourage them to use key app features. For example, you could send an in-app message to get users to subscribe, watch a video, complete a level, or buy an item. You can customize messages as cards, banners, modals, or images, and set up triggers so that they appear exactly when they'd benefit your users most.