

Corso di Sistemi Distribuiti e Cloud Computing

Corso di Laurea Magistrale in Ingegneria Informatica A.A. 2021/22
DIMES - Università degli Studi della Calabria



GOOGLE CLOUD PLATFORM

Google Cloud Platform

- È una suite di servizi di Cloud Computing che funziona sulla stessa infrastruttura che Google utilizza internamente per i suoi prodotti destinati all'utente finale, come Google Search e YouTube.
- Lanciato nel 2011, Google Cloud Platform è uno dei principali competitor di Amazon Web Services (AWS)
- Fornisce servizi su vasta scala, tra cui intelligenza artificiale e machine learning.



Google Cloud Platform

Google Cloud Platform

Compute



Compute Engine



App Engine



Container Engine



Container Registry



Cloud Functions

Identity & Security



Cloud IAM



Cloud Resource Manager



Cloud Security Scanner



Cloud Platform Security

Networking



Cloud Virtual Network



Cloud Load Balancing



Cloud CDN



Cloud Interconnect



Cloud DNS

Big Data



BigQuery



Cloud Dataflow



Cloud Dataproc



Cloud Datalab



Cloud Pub/Sub



Genomics

Storage and Databases



Cloud Storage



Cloud Bigtable



Cloud Datastore



Cloud SQL



Persistent Disk

Machine Learning



Cloud Machine Learning



Vision API



Speech API



Natural Language API



Translation API



Jobs API

Developer Tools



Cloud SDK



Deployment Manager



Cloud Source Repositories



Cloud Tools for Android Studio



Cloud Tools for IntelliJ



Cloud Tools for PowerShell



Cloud Tools for Visual Studio



Google Plug-in for Eclipse



Cloud Test Lab



Google Cloud Platform - Livello gratuito

- **Credito gratuito:** \$ 300 di credito gratuito per iniziare a usare un prodotto GCP per 12 mesi.
- **Always free:** limiti per l'utilizzo gratuito relativi ai prodotti inclusi nell'offerta per i clienti idonei, durante e dopo la prova gratuita di 12 mesi.

Google App Engine

28 ore istanza al giorno

5 GB di Cloud Storage

Memcache condivisa

1000 operazioni di ricerca al giorno, 10 MB per

l'indicizzazione delle ricerche

100 email al giorno

Google Cloud Firestore

1 GB di spazio di archiviazione

50.000 operazioni di lettura, 20.000 operazioni di scrittura, 20.000 operazioni di eliminazione al giorno

Google Compute Engine

1 istanza f1-micro al mese (solo aree geografiche degli Stati Uniti, a esclusione della Virginia del Nord [us-east4])

30 GB di HDD al mese

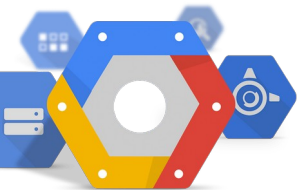
5 GB di snapshot al mese solo in determinate aree geografiche.

1 GB di traffico di rete in uscita dal Nord America verso destinazioni di qualsiasi area geografica (eccetto Cina e Australia) al mese



Google Cloud Platform - Livello gratuito

- **Google Cloud Storage:** 5GB al mese, 1GB di traffico mensile.
- **Google BigQuery:** ambiente gestito di data warehouse per analisi su scala di petabyte di dati. Limiti: 1TB query al mese, 10GB di archiviazione.
- **Google Natural Language Processing:** API per Google per l'estrazione di informazioni significative da testo non strutturato mediante algoritmi di machine learning. Limite: 5000 unità al mese.
- **Google Cloud Speech:** tecnologia per la trascrizione del parlato a testo (usato in Android e nelle applicazioni Google)
- **Google Cloud Vision:** rilevamento di etichette, OCR, rilevamento facciale e altro. Limite: 1000 unità al mese.



Google Cloud Platform

- Servizi di **calcolo** di Google Cloud Platform prevedono:
 - macchine virtuali
 - container (es. Docker)
 - gestione dei container (es. Kubernetes)
- Servizi di **storage** annoveriamo due grandi categorie:
 - archiviazione di file
 - archiviazione di dati (principalmente NoSQL)
- I servizi Big Data di Google Cloud Platform
- Servizi per la gestione e il processamento di **Big Data**:
 - pipeline di dati
 - data warehousing
 - machine learning



Google Kubernetes Engine

- **Google Kubernetes Engine (GKE)**, un ambiente Kubernetes per la gestione di container docker che offre funzionalità aggiuntive come autoscale, autorepair e aggiornamento automatico.
- Offre le più recenti innovazioni disponibili sul mercato apportando significativi vantaggi in termini di produttività degli sviluppatori.
- Supporta acceleratori hardware (es. GPU e HPC) per l'esecuzione di carichi di lavoro elevato (es. per applicazioni di machine learning).

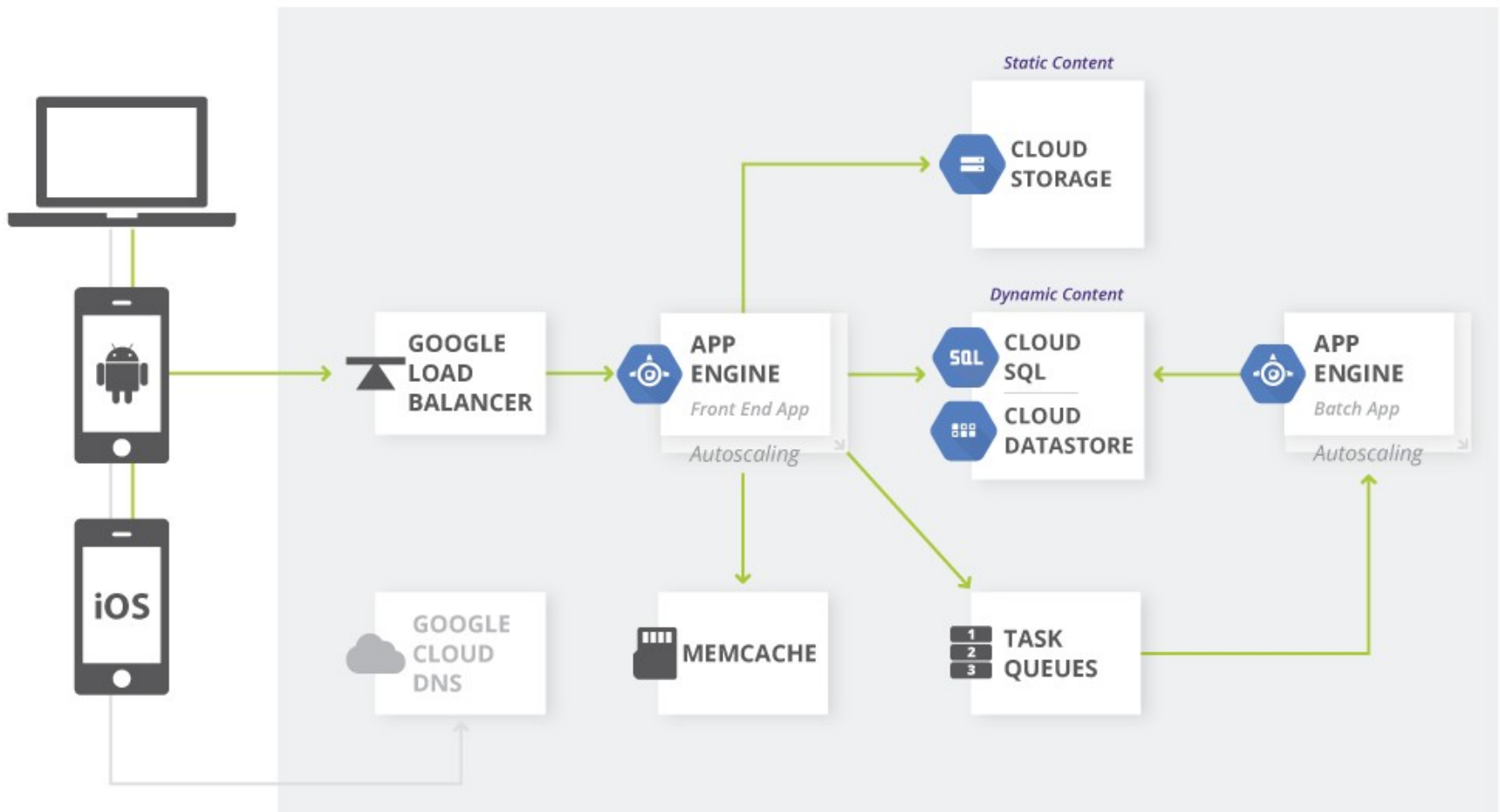


Google App Engine

- Google App Engine è una piattaforma di cloud computing, distribuita come servizio PaaS, per lo sviluppo e l'hosting di applicazioni web gestite dai Google Data Center
- E' una piattaforma interamente *managed*, in modo che gli sviluppatori debbano preoccuparsi solo del loro codice e non del funzionamento degli altri servizi.
- Supporta **Node.js, Java, Ruby, C#, Go, Python** e **PHP**.
- Strumenti di devops molto popolari come Chef, Puppet, Ansible, Salt, Docker, Consul e Swarm
- Consente lo sviluppo sia per il cloud, sia per ambienti *on-premises* (soluzioni installate sulla macchine dei singoli utenti).



Google App Engine



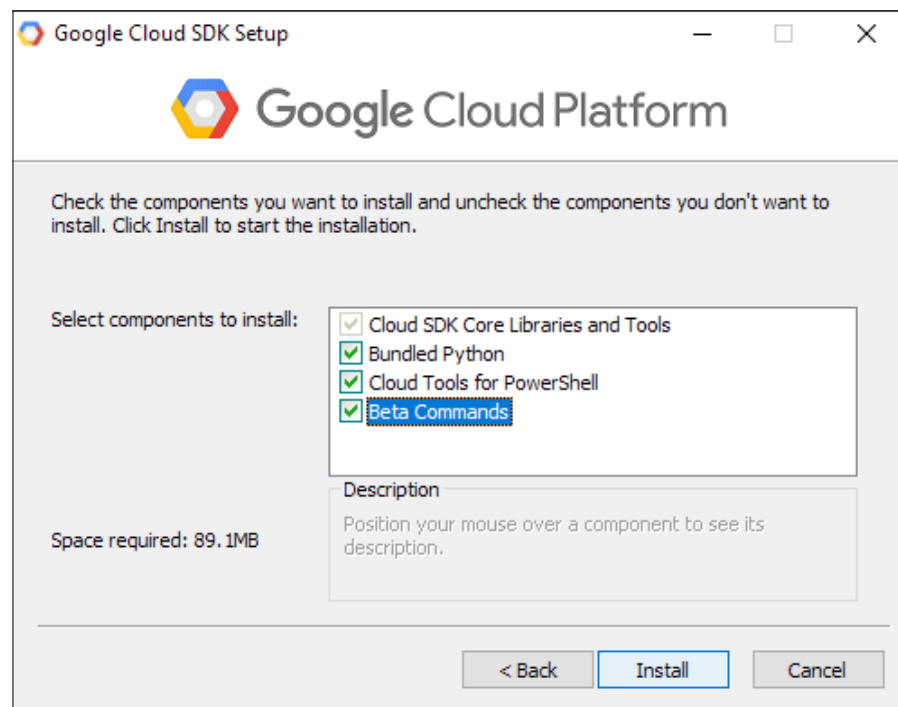
Quickstart for App Engine Standard Environment

- Scarica e installa Google Cloud SDK
- Successivamente verrà aperta una console dalla quale poter creare nuovi progetto o gestire progetti esistenti

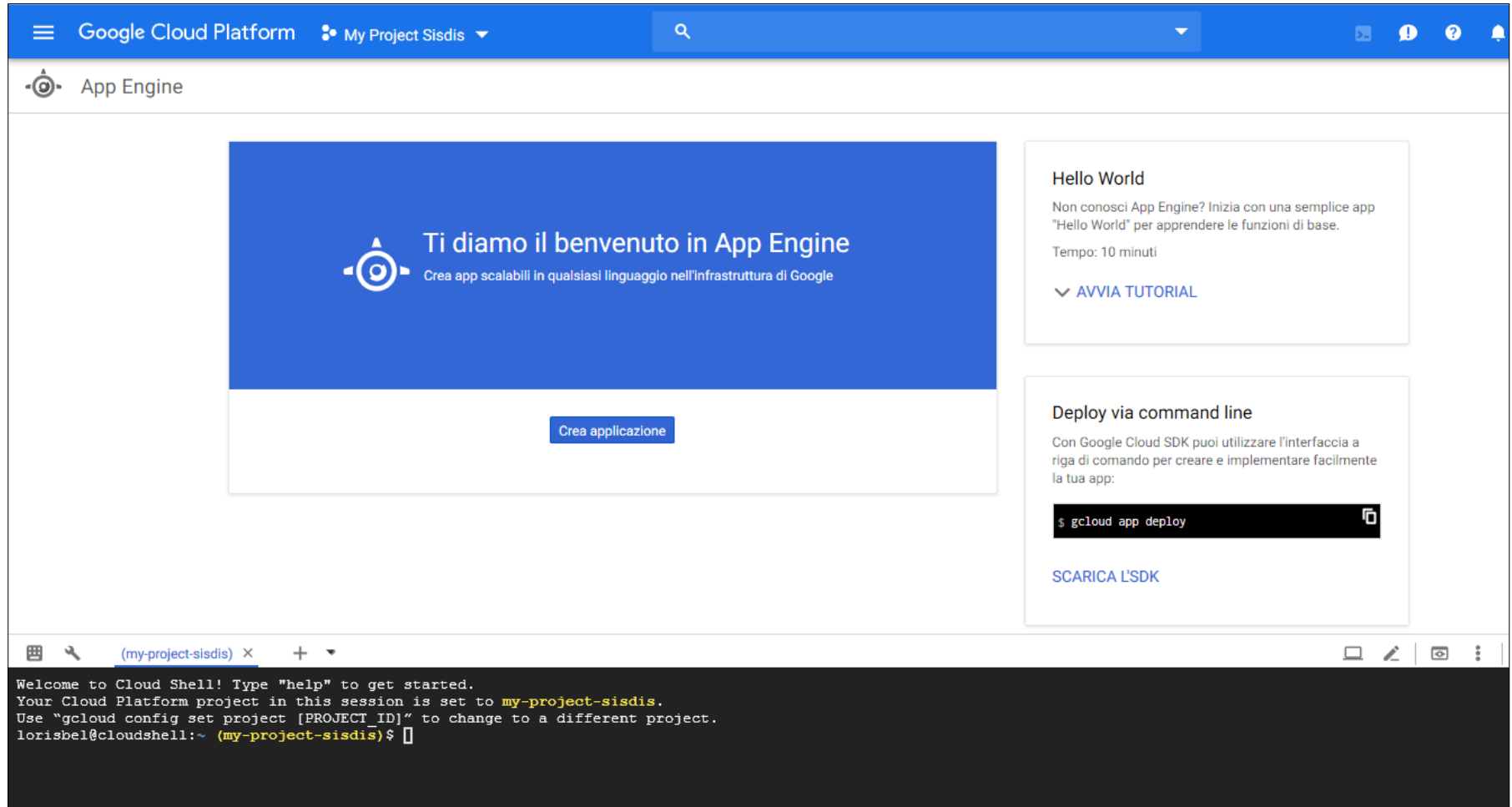
```
C:\Windows\SYSTEM32\cmd.exe - gcloud init

You are logged in as: [lorisbel@gmail.com].

Pick cloud project to use:
[1] angelic-tracer-239916
[2] my-project-sisdis
[3] roomours-1496939618384
[4] vast-watch-833
[5] Create a new project
Please enter numeric choice or text value (must exactly match list item):
```




Quickstart for App Engine Standard Environment



The screenshot shows the Google Cloud Platform interface for the App Engine Standard Environment. The top navigation bar includes the Google Cloud Platform logo, the project name "My Project Sisdis", a search bar, and notification icons. The main content area features a large blue banner with the App Engine logo and the text "Ti diamo il benvenuto in App Engine" (We welcome you to App Engine), followed by "Crea app scalabili in qualsiasi linguaggio nell'infrastruttura di Google" (Create scalable apps in any language on Google's infrastructure). A "Crea applicazione" (Create application) button is located below the banner. To the right, there are two instructional cards. The first card, titled "Hello World", explains that users can start with a simple "Hello World" app to learn the basics, with an estimated time of 10 minutes, and includes a link to "AVVIA TUTORIAL" (Start Tutorial). The second card, titled "Deploy via command line", describes using the Google Cloud SDK to create and implement an app via the command line, showing a terminal snippet with the command `$ gcloud app deploy`, and includes a link to "SCARICA L'SDK" (Download the SDK). At the bottom, a Cloud Shell terminal window is open, displaying a welcome message and instructions on how to set the project ID, with the current project ID being "my-project-sisdis".

Google Cloud Platform My Project Sisdis

App Engine

 **Ti diamo il benvenuto in App Engine**
Crea app scalabili in qualsiasi linguaggio nell'infrastruttura di Google

[Crea applicazione](#)

Hello World
Non conosci App Engine? Inizia con una semplice app "Hello World" per apprendere le funzioni di base.
Tempo: 10 minuti
[AVVIA TUTORIAL](#)

Deploy via command line
Con Google Cloud SDK puoi utilizzare l'interfaccia a riga di comando per creare e implementare facilmente la tua app:

```
$ gcloud app deploy
```


[SCARICA L'SDK](#)

(my-project-sisdis) x + ▾

```
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to my-project-sisdis.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
lorisbel@cloudshell:~ (my-project-sisdis) $
```



Quickstart for App Engine Standard Environment


Google Cloud Platform My Project Sisdis

App Engine Crea applicazione

Region

L'area geografica è permanente.

europa-west



Google Map data ©2019

Crea applicazione Annulla

Google Cloud Platform My Project Sisdis

App Engine Inizia

Questo passaggio è facoltativo. Il suo scopo è guidarti agli esempi di codice e SDK

Language

Java

Environment

Ogni versione della tua app può utilizzare il runtime standard o flessibile. Puoi cambiarlo in seguito.

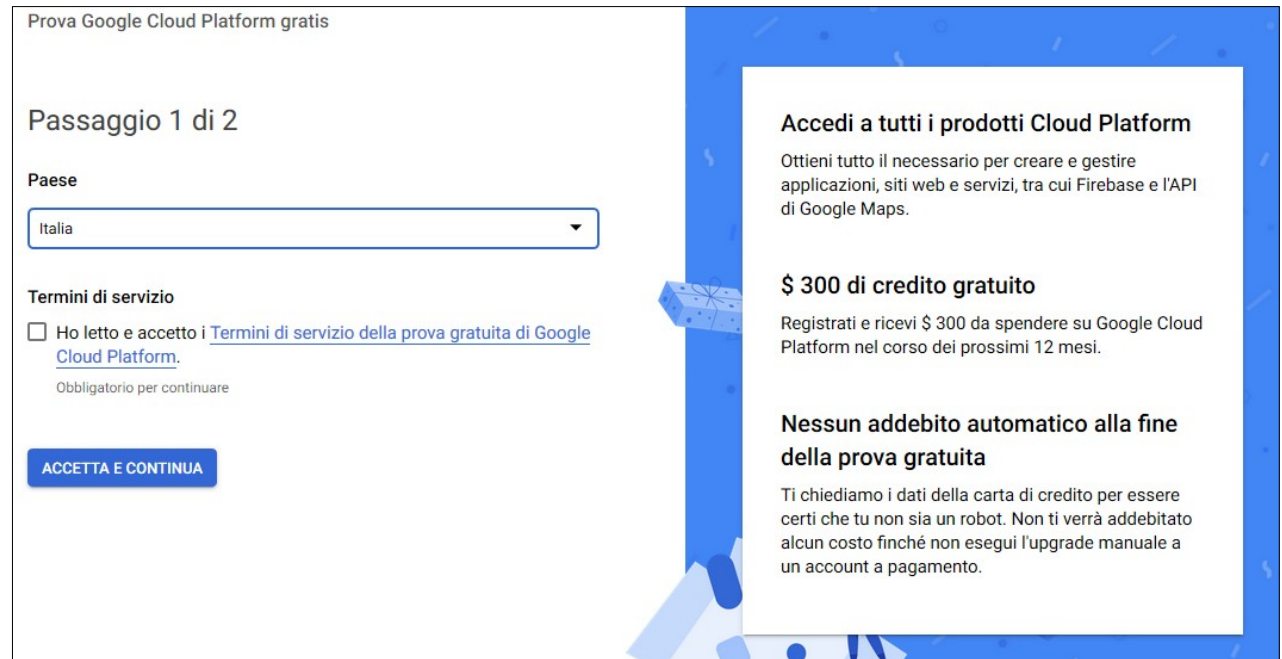
Standard

Avanti Annulla



Quickstart for App Engine Standard Environment

- Verrà richiesta una registrazione per con carta di credito per verificare che non si tratta di un robot.
- Vengono messi a disposizione \$300 entro 12 mesi su Google Cloud Platform.
- Non ti verrà addebitato alcun costo finché non esegui l'upgrade manuale a un account a pagamento.



Prova Google Cloud Platform gratis

Passaggio 1 di 2

Paese

Italia

Termini di servizio

☐ Ho letto e accetto i [Termini di servizio della prova gratuita di Google Cloud Platform.](#)

Obbligatorio per continuare

ACCETTA E CONTINUA

Accedi a tutti i prodotti Cloud Platform

Ottieni tutto il necessario per creare e gestire applicazioni, siti web e servizi, tra cui Firebase e l'API di Google Maps.

\$ 300 di credito gratuito

Registrati e ricevi \$ 300 da spendere su Google Cloud Platform nel corso dei prossimi 12 mesi.

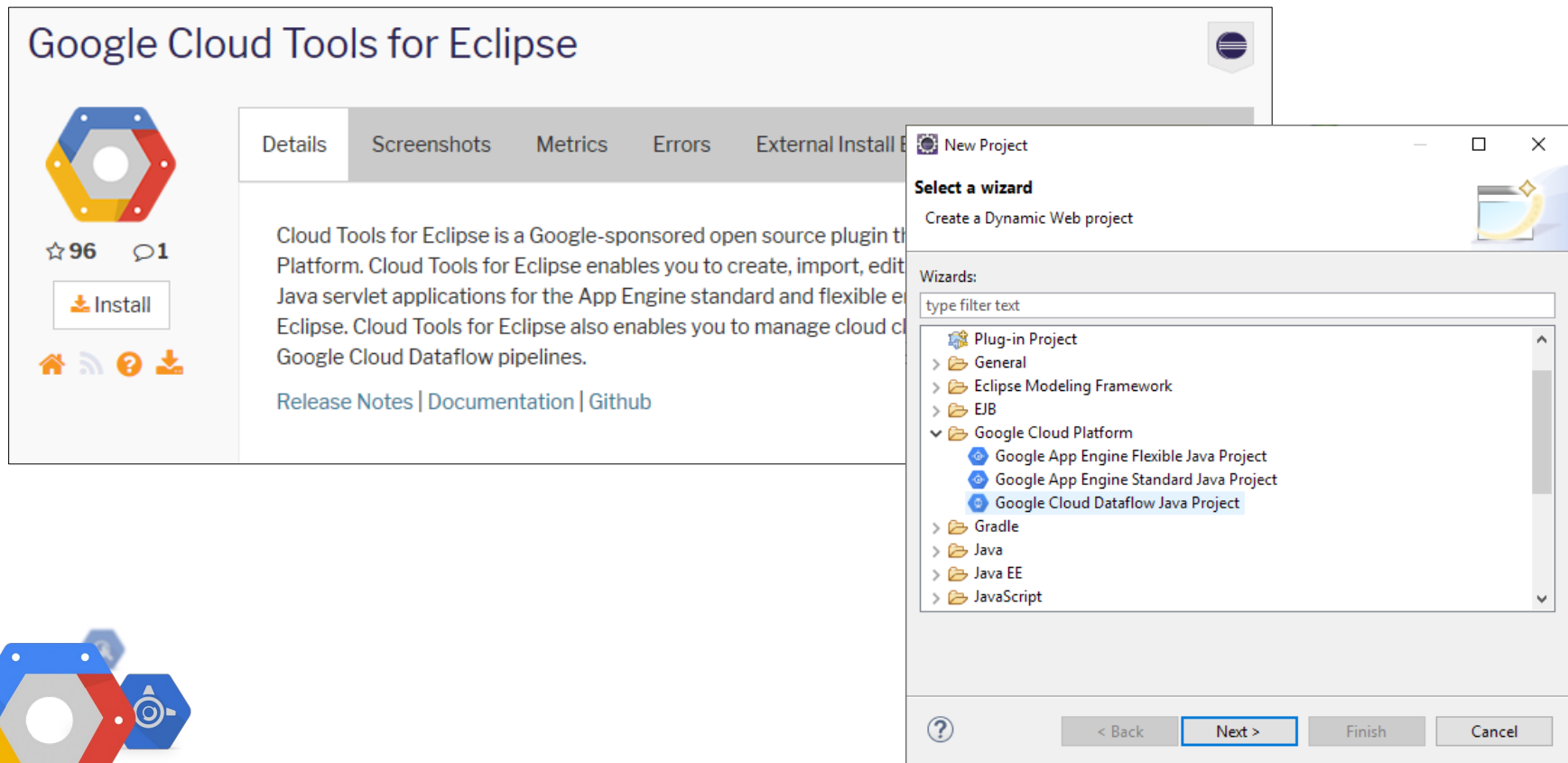
Nessun addebito automatico alla fine della prova gratuita

Ti chiediamo i dati della carta di credito per essere certi che tu non sia un robot. Non ti verrà addebitato alcun costo finché non esegui l'upgrade manuale a un account a pagamento.



Quickstart for App Engine Standard Environment

- E' possibile gestire i propri progetti direttamente da Eclipse tramite l'installazione di Google Cloud Tools for Eclipse.
- <https://marketplace.eclipse.org/content/google-cloud-tools-eclipse>



The image shows a screenshot of the Eclipse IDE interface. On the left, the Eclipse Marketplace displays the 'Google Cloud Tools for Eclipse' plugin. The plugin's details are visible, including its description: 'Cloud Tools for Eclipse is a Google-sponsored open source plugin that enables you to create, import, edit, and run Java servlet applications for the App Engine standard and flexible environments from Eclipse. Cloud Tools for Eclipse also enables you to manage cloud resources and Google Cloud Dataflow pipelines.' Below the description are links for 'Release Notes', 'Documentation', and 'Github'. An 'Install' button is also present.

On the right, the 'New Project' wizard is open. The 'Select a wizard' dialog shows a list of available project types. Under the 'Google Cloud Platform' category, the following options are listed:

- Google App Engine Flexible Java Project
- Google App Engine Standard Java Project
- Google Cloud Dataflow Java Project

The 'Google App Engine Standard Java Project' option is selected. The wizard has a 'Next >' button highlighted, indicating the next step in the process.

Quickstart for App Engine Standard Environment

- Le applicazioni create in Eclipse possono essere testate su server locali (es. usando Apache Tomcat con application server) o direttamente sui server di App Engine.

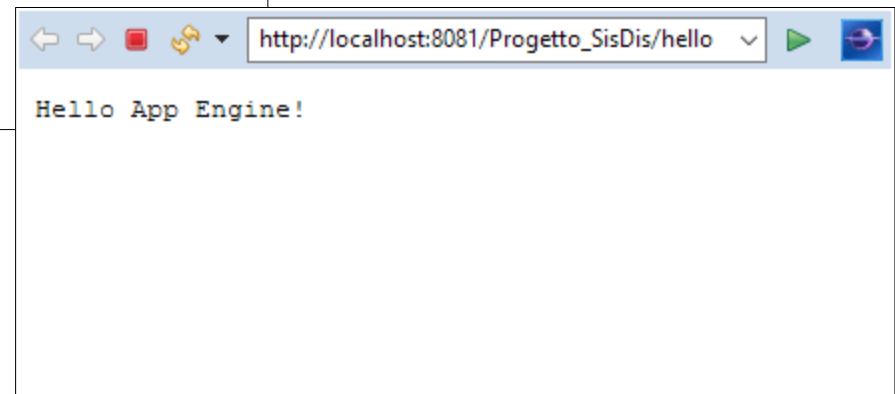
```
import java.io.IOException;

@SuppressWarnings("serial")
@WebServlet(
    name = "HelloAppEngine",
    urlPatterns = {"/hello"}
)
public class HelloAppEngine extends HttpServlet {

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException {

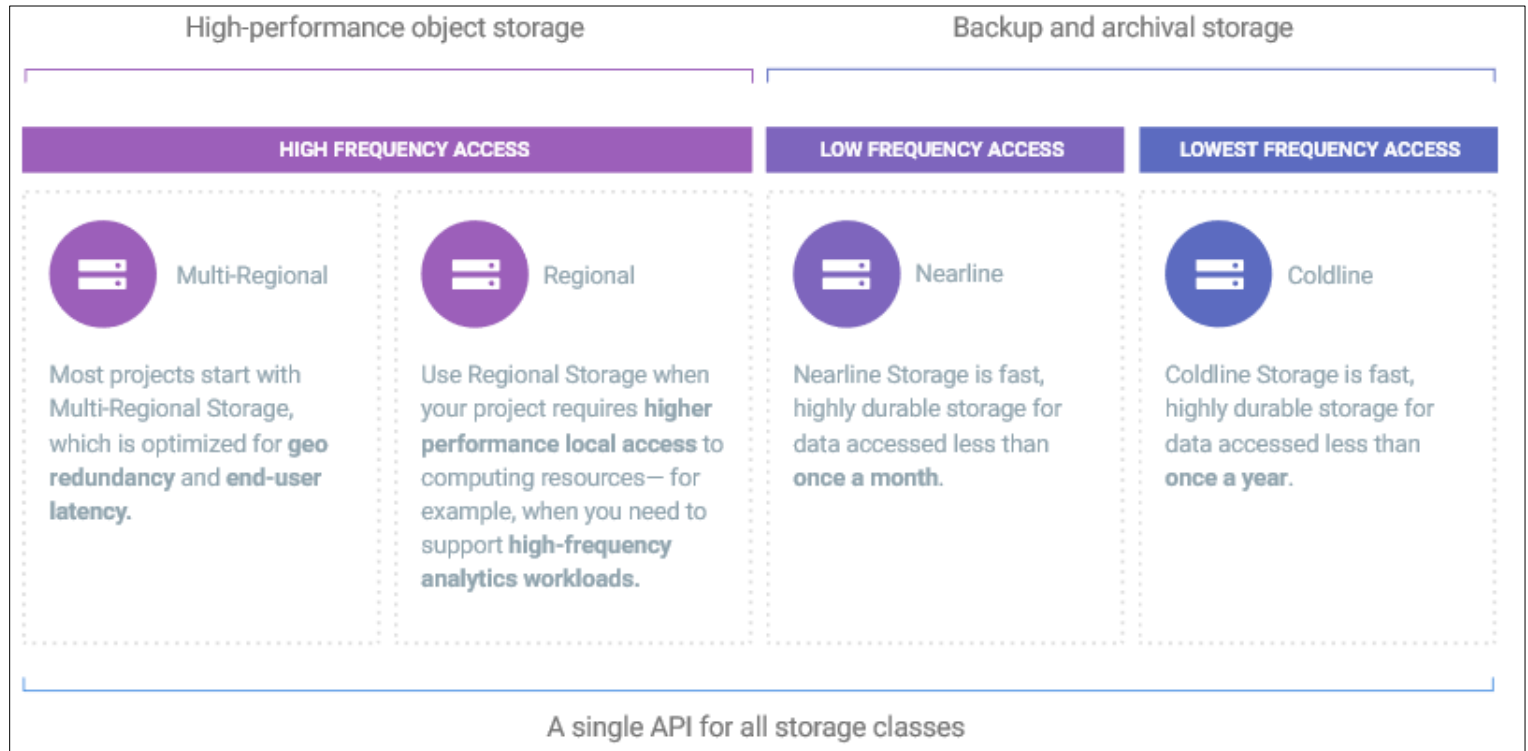
        response.setContentType("text/plain");
        response.setCharacterEncoding("UTF-8");

        response.getWriter().print("Hello App Engine!\r\n");
    }
}
```



Google Cloud Storage

- **Google Cloud Storage** è una soluzione di archiviazione di file destinata agli sviluppatori e agli amministratori di sistema
- Ha API che consentono alle app di archiviare e recuperare oggetti.
- È dotato di cache globale, versioning, OAuth e controlli di accesso granulari, con livelli di sicurezza configurabili.



Google Cloud Datastore

- **Google Cloud Datastore** è un database NoSQL a scalabilità elevata per le applicazioni.
- Gestisce automaticamente il partizionamento orizzontale e la replica su un numero elevato di nodi
- Fornisce elevati livelli di disponibilità e durabilità, in grado di scalare automaticamente per gestire il carico delle applicazioni.
- Offre una miriade di funzionalità come transazioni ACID (in modo che le operazioni raggruppate abbiano tutte esito positivo o negativo), query di tipo SQL (linguaggio GQL), indicizzazione dei dati.
- **Interfaccia RESTful** per accedere ai dati in formato JSON da qualsiasi destinazione del deployment.
- Possibilità di usare diversi client open source o gli ORM (*Object-Relational Mapping*) gestiti dalla community (*Objectify*, *NDB*).
- È possibile creare soluzioni che si estendono su **App Engine e Compute Engine** e si basano su Cloud Datastore come punto di integrazione.



Google Firebase

- **Capacità di sincronizzazione dei dati:** Firebase è in grado di aggiornare i dati istantaneamente, sia se integrato in app web che mobile. Non appena l'app recupera la connettività sincronizza i dati mostrati con le ultime modifiche apportate;
- **Integrazione** mediante disponibilità di librerie client per integrare Firebase in ogni app Android, Javascript e framework annessi (es. Node.js, Angular.js, Ember.js, Backbone.js), Java e sistemi Apple.
- **API REST** per consentire l'integrazione con sistemi per i quali non esistono librerie apposite.
- **Sicurezza:** i dati immagazzinati in Firebase sono replicati e sottoposti a backup continuamente. La comunicazione con i client avviene sempre in modalità crittografata tramite SSL con certificati a 2048-bit.



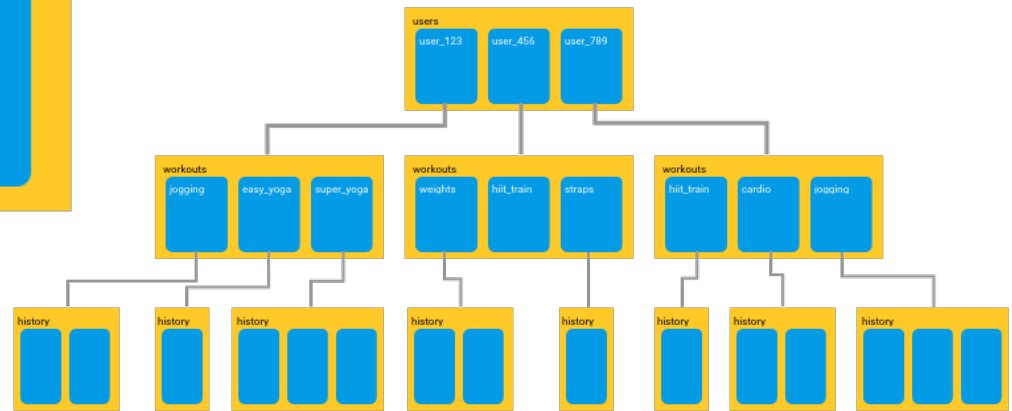
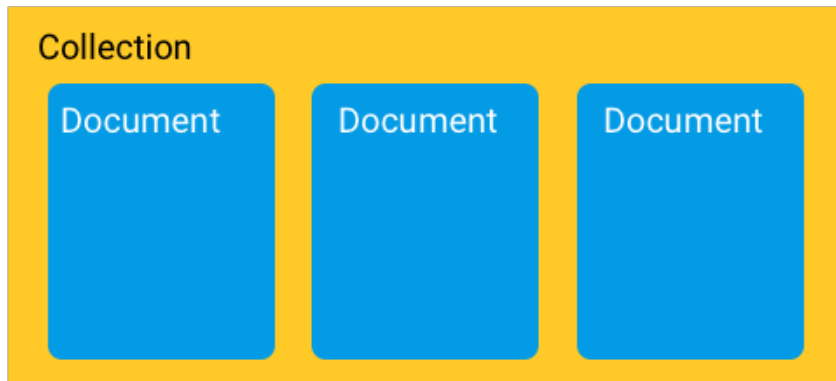
Google Firebase - Realtime database vs Firestore

- Firebase offre due tipi di soluzioni di storage: **Realtime Database** e **Cloud Firestore**.
- **Realtime Database** è il primo e originale database basato su cloud di Firebase, creato per le app mobili che richiedono stati sincronizzati tra i client in tempo reale. E' una soluzione efficiente e a bassa latenza.
- **Cloud Firestore** è il nuovo database di punta di Firebase per le app mobili. È un successore del Realtime Database con un nuovo e più intuitivo modello di dati. Fornisce maggiori funzionalità, è più veloce e più scalabile del Realtime Database.



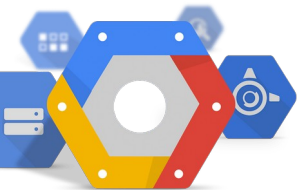
Cloud Firestore

- **Migliori query e dati più strutturati:** mentre Realtime Database è in pratica un albero JSON di grandi dimensioni, Firestore è molto più strutturato: è un database *document-based*, il che significa che tutti i tuoi dati sono archiviati in oggetti chiamati documenti costituiti da coppie chiave-valore e questi valori possono contenere qualsiasi cosa (string, float, dati binari, json).
- I documenti sono organizzati in collection. Ogni documento può essere collegato ad una sotto-collection.



Cloud Firestore - Altre differenze

- **Multi-Region support for better reliability:** Cloud Firestore supporta storage multi-regione. Questo significa che i dati sono automaticamente copiati su datacenter multipli in regioni geograficamente distanti.
- **Different pricing model:** Realtime Database calcola il costo sulla base della quantità di dati memorizzati nel database; Firestore, invece, principalmente sul numero di operazioni di lettura/scrittura che vengono eseguite.
- **Better scalability:** Realtime Database è in grado di supportare fino a circa 100,000 connessioni concorrenti, mentre Cloud Firestore riesce a gestirne fino a 1,000,000.
- **Better querying:** Cloud Firestore ha un sistema di querying più avanzato di quello di Realtime Database. Con Realtime Database, infatti, creare query complesse su campi multipli diventa spesso macchinoso e complicato.



Google Firebase - Vantaggi

- **Realtime:** Firebase è un real-time cloud-based database che permette di memorizzare i dati JSON e di sincronizzarli continuamente con tutti i client connessi.
 - Esempio: app che fornisce aggiornamenti in tempo reale agli utenti senza creare API apposite.
- **Authentication:** Firebase fornisce librerie e SDK per l'autenticazione del client attraverso l'applicazione utilizzando varie tecniche.
- **Storage:** Firebase è molto utile per creare app per la memorizzazione e la distribuzione di file agli utenti (video, immagini, etc.).
- **Notifiche:** Firebase fornisce un servizio gratuito per le notifiche, molto utile in caso di app mobile. E' possibile usare una GUI console per creare/inviare notifiche verso utenti specifici.
- **App Indexing:** questa funzione viene utilizzata per indicizzare l'applicazione nei risultati di ricerca di Google.



Google Firebase - Struttura del database

- I dati all'interno del Realtime Database sono memorizzati come oggetti JSON.
- Rispetto ai db SQL, non ci sono tabelle o record, ma sono alberi JSON di informazioni.
- L'aggiunta di dati all'albero JSON corrisponde alla creazione di un nodo dell'albero, con struttura JSON, a cui è associate una chiave.
- Le chiavi possono essere definite dall'utente (es. ID utente) o autogenerate (es., valore incrementale in seguito ad operazione di push).

```
{
  "users": {
    "mariobianchi": {
      "name": "Mario Bianchi",
      "contacts": { "sergiorossi":
true },
    },
    "sergiorossi": { ... },
    "ugoneri": { ... }
  }
}
```



Google Firebase

Database Realtime Database

Dati Regole Backup Utilizzo

<https://my-project-sisdis.firebaseio.com/>

```
my-project-sisdis
├── users
│   ├── mariobianchi
│   │   ├── contacts
│   │   │   └── sergiorossi: true
│   │   └── name: "Mario Bianchi"
│   ├── sergiorossi
│   │   └── name: "Sergio Rossi"
│   └── ugoneri
│       └── name: "Ugo Neri"
```



Google Firebase Cloud Notification

- **Firebase Cloud Messaging** è una soluzione di messaggistica multiplatforma che ti consente di inviare messaggi in modo affidabile senza alcun costo.
- Consente di notificare a un'applicazione client (mobile o desktop) che sono disponibili nuove e-mail o altri dati da sincronizzare.
- È possibile inviare messaggi di notifica per facilitare il ritorno e la fidelizzazione degli utenti.
- Per casi d'uso come la messaggistica istantanea, un messaggio può trasferire un carico utile fino a 4KB in un'applicazione client.



Google Firebase Example

- Creamo un nuovo Progetto da <https://console.firebase.google.com>



Google Firebase Example

- Dalle impostazioni del progetto, andare su Account di Servizio e scaricare un file json contenente una chiave privata da usare nell'app per connettersi ad database Firebase.

The screenshot shows the Google Firebase console interface. On the left is a dark sidebar with the 'Firebase' logo and a list of project tools including 'Panoramica del progetto', 'Sviluppo', 'Qualità', 'Analisi', 'Crescita', 'Extensions', and 'Spark'. The main area is titled 'Settings' for the project 'Demo CRUD Sistemi Distribuiti'. Below the title are tabs for 'Generale', 'Cloud Messaging', 'Integrazioni', 'Account di servizio', 'Privacy dei dati', and 'Utenti e autorizzazioni'. The 'Account di servizio' tab is selected and circled in red. It contains a section for 'SDK di Firebase Admin' with a 'Genera nuova chiave privata' button also circled in red. The page also displays the current service account email and a code snippet for initializing the Firebase Admin SDK in Node.js.

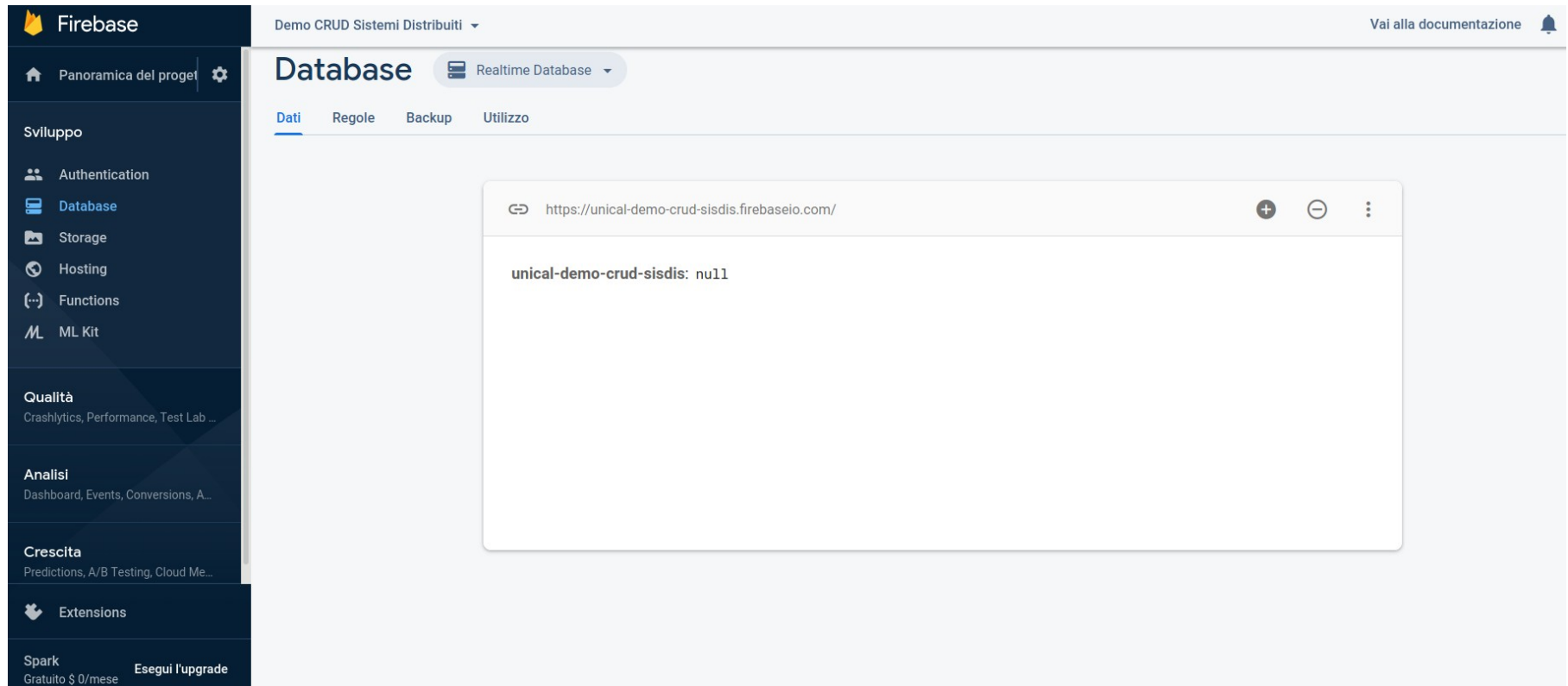
```
var admin = require("firebase-admin");

var serviceAccount = require("path/to/serviceAccountKey.json");

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: "https://unical-demo-crud-sisdis.firebaseio.com"
});
```

Google Firebase Example

- Ora andiamo su Sviluppo → Database → Crea Realtime Database



Google Firebase Example

- Ora creiamo una semplice applicazione Java in grado di leggere e scrivere sul database Firestore.
- E' necessario importare la libreria Firebase Admin Java SDK (es. usando Maven)

```
<dependencies>
  <dependency>
    <groupId>com.google.firebase</groupId>
    <artifactId>firebase-admin</artifactId>
    <version>6.13.0</version>
  </dependency>
</dependencies>
```

Google Firebase Example

- Creiamo una classe per gestire il collegamento al database Firestore.

```
public class FirebaseService {  
    FirebaseDatabase db;  
  
    public FirebaseService(String keyPath, String dbUrl) throws IOException {  
        FileInputStream serviceAccount =  
            new FileInputStream(keyPath);  
  
        FirebaseOptions options = new FirebaseOptions.Builder()  
            .setCredentials(GoogleCredentials.fromStream(serviceAccount))  
            .setDatabaseUrl(dbUrl)  
            .build();  
  
        FirebaseApp.initializeApp(options);  
        db = FirebaseDatabase.getInstance();  
    }  
  
    public FirebaseDatabase getDb() {  
        return db;  
    }  
}
```

Google Firebase Example

- Creiamo una classe Contact per rappresentare un contatto da inserire nel database.

```
public class Contact implements Serializable {
    private String id, name, surname, address, phone;
    public Contact() {}
    public Contact(String line) {
        JSONObject json = new JSONObject(line);
        this.id = json.getString("id");
        this.name = json.getString("name");
        this.phone = json.getString("phone");
        this.address = json.getString("address");
    }

    // Metodi Getter e setter
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getAddress() { return address; }
    public void setAddress(String address) { this.address = address; }
    public String getPhone() { return phone; }
    public void setPhone(String phone) { this.phone = phone; }

    @Override
    public String toString() {
        return new Gson().toJson(this).toString();
    }
}
```

Google Firebase Example

- Creiamo un classe ImportDbUsers, che implementa Runnable, per importare in blocco degli utenti sul database.

```
public class ImportDbusers implements Runnable {  
  
    private final List<Contact> users;  
    private FireBaseService fbs;  
  
    public ImportDbusers(FireBaseService fbs, List<Contact> users) {  
        this.fbs = fbs;  
        this.users = users;  
    }  
  
    public void run() {  
        DatabaseReference ref = fbs.getDb().getReference("/users/");  
        ref.setValueAsync(users);  
    }  
}
```


Google Firebase Example

- Creiamo un classe ShowDbChanges, che implementa Runnable, per rilevare mediante eventi le modifiche che avvengono sul database.

```
public class ShowDbChanges implements Runnable {  
  
    private FireBaseService fbs;  
  
    public ShowDbChanges(FireBaseService fbs) { this.fbs = fbs; }  
  
    public void run() {  
        DatabaseReference ref = fbs.getDb().getReference("/");  
  
        ref.addValueEventListener(new ValueEventListener() {  
            public void onDataChange(DataSnapshot dataSnapshot) {  
                Object document = dataSnapshot.getValue();  
                System.out.println(document);  
            }  
            public void onCancelled(DatabaseError error) {  
                System.out.print("Error: " + error.getMessage());  
            }  
        });  
    }  
}
```

Google Firebase Example

- Creiamo un classe MakeDbChanges, che implementa Runnable, per scrivere in maniera asincrona sul database.

```
public class MakeDbChanges implements Runnable {  
  
    private final int size;  
    private FireBaseService fbs;  
    private Random rand = new Random();  
  
    public MakeDbChanges(FireBaseService fbs, int size) {  
        this.fbs = fbs; this.size = size;  
    }  
  
    public void run() {  
        while(true) {  
            try {  
                DatabaseReference ref = fbs.getDb()  
                    .getReference("/users/"+rand.nextInt(size)+"/modifiedAt");  
                ref.setValueAsync(new Date());  
                Thread.sleep(2000);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

Google Firebase Example

- Creiamo, infine, una class main per lanciare l'applicazione.

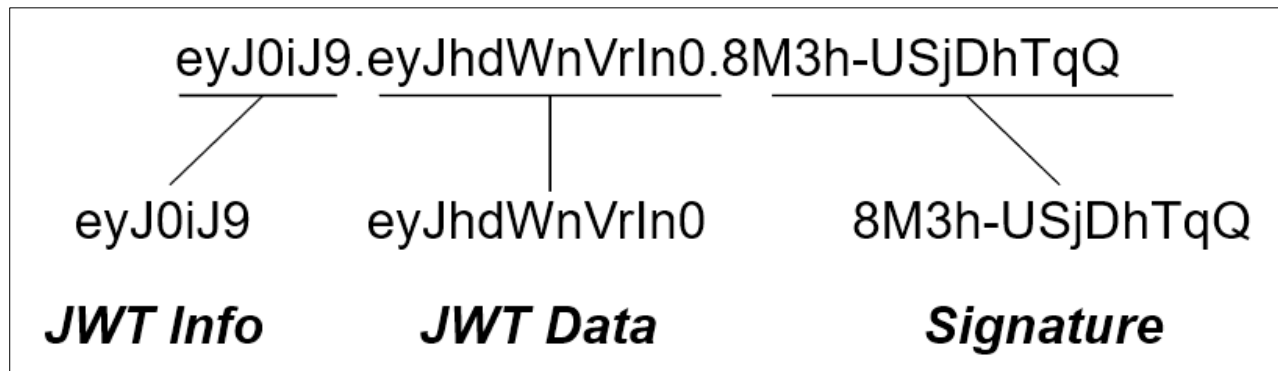
```
public class Main {  
    public static void main(String[] args) throws IOException {  
        FireBaseService fbs = null;  
        try {  
            String keyPath = "/home/loris/serviceAccountKey.json";  
            String dbUrl = "https://unical-demo-crud-sisdis.firebaseio.com";  
            fbs = new FireBaseService(keyPath, dbUrl);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
  
        List<Contact> users = Files.lines(Paths.get("usersDemo.json"))  
            .map(line -> new Contact(line)).collect(Collectors.toList());  
  
        Thread t = new Thread(new ShowDbChanges(fbs));  
        Thread t2 = new Thread(new ImportDbusers(fbs, users));  
        Thread t3 = new Thread(new MakeDbChanges(fbs, users.size()));  
  
        t.run(); t2.run(); t3.run();  
  
        try {  
            // Aspettiamo perchè le operazioni sono asincrone  
            Thread.sleep(100000);  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

JSON Web Token (JWT)

- JWT è un mezzo per inviare un messaggio a terzi in modo tale che il destinatario possa validare chi lo ha inviato.
- Quando una terza parte riceve un messaggio questo potrebbe contenere, ad esempio nell'header, una stringa JWT.
- Chi ha ricevuto il messaggio può ottenere la chiave pubblica dei mittenti e utilizzarla per convalidare la firma JWT.
- Se la firma è valida, il JWT deve essere stato firmato con la chiave privata corrispondente, quindi deve provenire dal mittente previsto.
- Consente di notificare a un'applicazione client (mobile o desktop) che sono disponibili nuove e-mail o altri dati da sincronizzare.

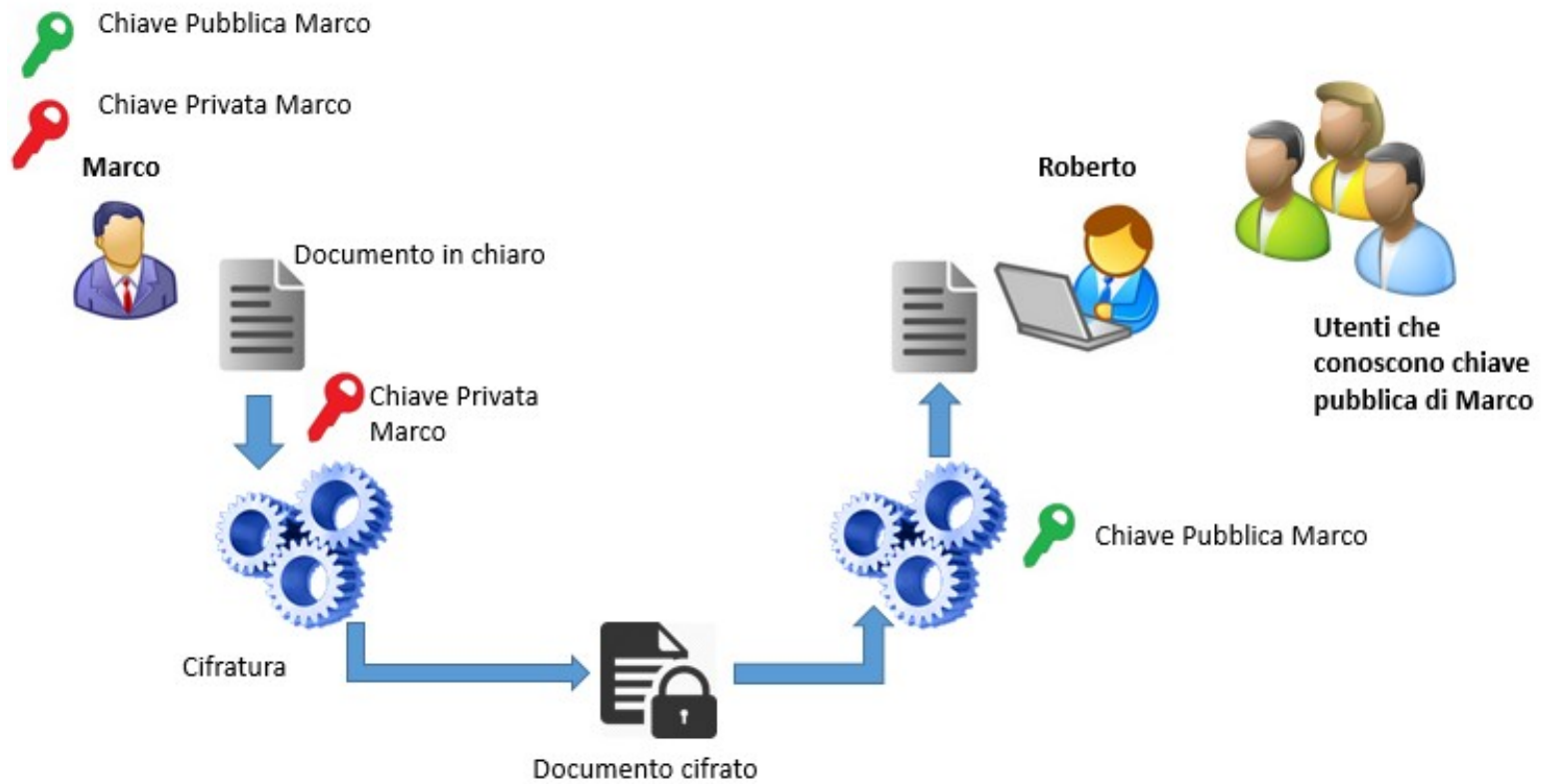
JSON Web Token (JWT)

- Un JWT firmato è solo una stringa, sebbene possa essere pensato come tre stringhe unite da punti.
- La prima stringa (**JWT Info**) e la seconda stringa (**JWT Data**) sono pezzi di JSON codificati in **base64**, il che significa che è leggibile pubblicamente.
- JWT Info** fornisce informazioni sul JWT stesso, indicando quale algoritmo è stato usato per creare la firma.
- JWT Data** fornisce informazioni sul mittente, sul destinatario e sul tempo di scadenza del token.
- La terza parte del JWT (**Signature**) consente di verificare l'integrità dei dati contenuti nel token.



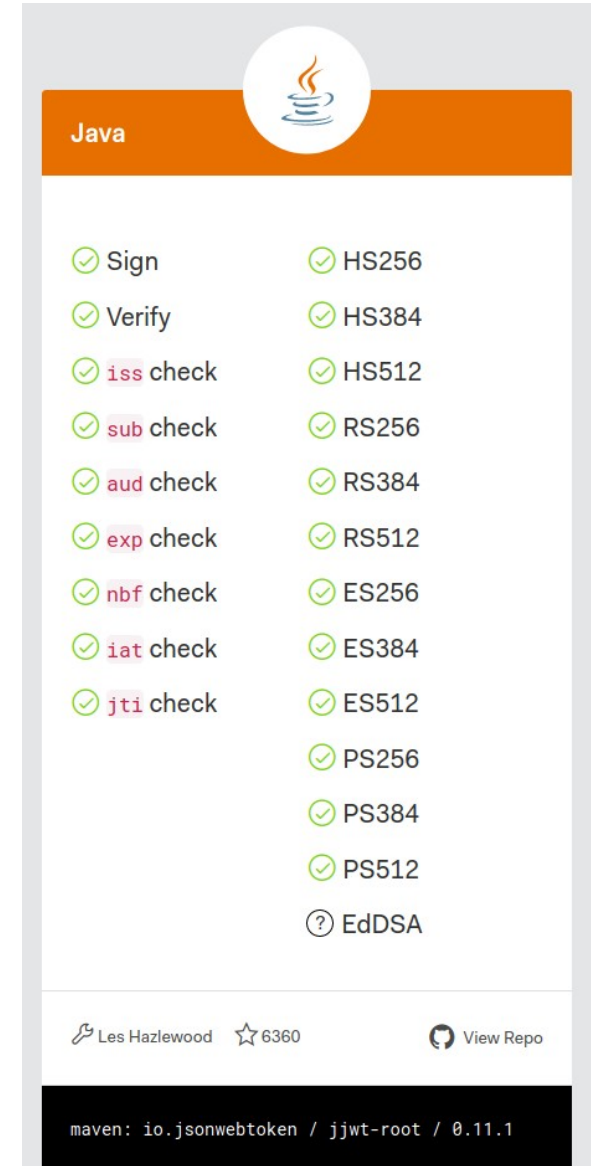
JSON Web Token (JWT)

- Quando i token vengono **firmati** utilizzando coppie di chiavi pubbliche e private, la firma valida anche il mittente, in quanto certifica che solo la parte che detiene la chiave privata è quella che l'ha firmata (il messaggio potrà essere decifrato solo con la chiave pubblica del mittente).



JSON Web Token (JWT)

- <https://github.com/jwtk/jjwt>
- Java JWT: JSON Web Token for Java and Android
- E' una libreria abbastanza semplice da usare e da comprendere per la creazione e la verifica di token Web JSON (JWT) su JVM e Android.
- Un elenco aggiornato delle migliori librerie per JWT per diversi linguaggi è disponibile al sito <https://jwt.io/>



Esempio JWT in Java

```
public class JWTMain {  
  
    // Get RSA keys. Uses key size of 2048.  
    private static KeyPair getRSAKeys() throws Exception {  
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");  
        keyPairGenerator.initialize(2048);  
        return keyPairGenerator.generateKeyPair();  
    }  
  
    ...  
}
```


Esempio JWT in Java

//Metodo per costruire un JWT

```
public static String createJWT(String id, String issuer, String subject, long ttlMillis, PrivateKey  
privateKey) {
```

//Algoritmo usato per generare la firma del JWT

```
SignatureAlgorithm signatureAlgorithm = SignatureAlgorithm.RS512;
```

```
long nowMillis = System.currentTimeMillis();  
Date now = new Date(nowMillis);  
JwtBuilder builder = Jwts.builder().setId(id)  
    .setIssuedAt(now)  
    .setSubject(subject)  
    .setIssuer(issuer)  
    .signWith(signatureAlgorithm, privateKey);
```

//Si aggiunge anche una data di scadenza del JWT

```
if (ttlMillis >= 0) {  
    long expMillis = nowMillis + ttlMillis;  
    Date exp = new Date(expMillis);  
    builder.setExpiration(exp);  
}
```

//Crea il JWT e lo serializza in una stringa URI-safe

```
return builder.compact();  
}
```

Esempio JWT in Java

```
public static Claims decodeJWT(String jwt, PublicKey publicKey) {  
    //Viene sollevata un'eccezione se il JWT non è correttamente firmato  
    Claims claims = Jwts.parser()  
        .setSigningKey(publicKey)  
        .parseClaimsJws(jwt).getBody();  
    return claims;  
}  
  
public static String keyPairToString(KeyPair keys) {  
    StringBuilder sb = new StringBuilder();  
    try {  
        KeyFactory fact = KeyFactory.getInstance("RSA");  
        PKCS8EncodedKeySpec spec = fact.getKeySpec(keys.getPrivate(),  
            PKCS8EncodedKeySpec.class);  
        X509EncodedKeySpec specPub = fact.getKeySpec(keys.getPublic(),  
            X509EncodedKeySpec.class);  
        sb.append("*** PRIVATE KEY \n"  
+Base64.getEncoder().encodeToString(spec.getEncoded()));  
        sb.append("*** PUBLIC KEY \n"  
+Base64.getEncoder().encodeToString(specPub.getEncoded()));  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return sb.toString();  
}  
...
```

Esempio JWT in Java

```
public static void main(String[] args) throws Exception {
    System.out.println("Generazione coppia chiavi RSA...");
    KeyPair rsaKeysUser = getRSAKeys();
    System.out.println(keyPairToString(rsaKeysUser));

    KeyPair rsaKeysUserFake = getRSAKeys();
    System.out.println("\n"+keyPairToString(rsaKeysUserFake));

    String jwtId = "1234567890";
    String jwtIssuer = "Sistemi Distribuiti JWT";
    String jwtSubject = "Pippo Rossi";
    int jwtTimeToLive = 600000;
    System.out.println("\nGenerazione JWT Token...");
    String jwt = JWTMain.createJWT(
        jwtId, jwtIssuer, jwtSubject, jwtTimeToLive,
        rsaKeysUser.getPrivate()
    );
    System.out.println(jwt);

    System.out.println("\nDecodifica JWT Token con Public Key valida...");
    Claims claims = JWTMain.decodeJWT(jwt, rsaKeysUser.getPublic());
    System.out.println(claims);

    System.out.println("\nDecodifica JWT Token con Public Key Fake...");
    claims = JWTMain.decodeJWT(jwt, rsaKeysUserFake.getPublic());
    System.out.println(claims);
}
```

JWT Decoder online

www.jsonwebtoken.io

JWT String

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJxMjM0NTY3ODkwIiwiaWF0IjoxNTkwMzUwOTc1LCJzdWllOiJQaXBwbyBSb3NzaSIsImZycyI6IiJldGkgZGkgQ2FsY29sYXRvcmkkgLSBEZW1vIEpXVCIsImV4cCI6MTU5MDM1NDY3NX0.q89sw1WGdxlp-4CfJB0rXUGVex-E0hiPhEkwZppnrjc
```

Header

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}
```

Payload

```
{  
  "jti": "1234567890",  
  "iat": 1590350975,  
  "sub": "Pippo Rossi",  
  "iss": "Reti di Calcolatori - Demo JWT",  
  "exp": 1590354640  
}
```

Signing Key

Verified

```
oeRaYY7Wo24sDqKSX3IM9ASGmdGPmkTd9jo1QTy4b7P9Ze5_9hKolVX8xNrQDcNRFEdTZNOuOyqEGhXEbdJI-ZQ19k_o9MI0y3eZN2lp9jow55FfXMiINEdt1XR85VipRLSOKT6kSpzs2
```