

**ЛЕКЦІЯ 5. ЗБЕРЕЖЕННЯ І
ВИДОБУВАННЯ ДАНИХ**
Глибовець А.М.

ВСТУП

- Сьогодні ми поговоримо про збереження даних
- Які способи збереження даних ви знаєте?



ВСТУП

- В Google App Engine є багато варіантів збереження даних, наприклад:
 - Cloud SQL
 - Cloud Storage
 - Cloud Datastore
- Ми розберемося з Datastore



DATASTORE

- Datastore – це база даних, що працює в Google Cloud
- Вона доступна для будь-якого App Engine застосування
- Це Big Table
 - Докладніше про Big Table
<http://research.google.com/archive/bigtable.html>
- Це сховище виду key-value (дуже схоже на HashTable)
- Також вона column-oriented (колонки, а не рядки зберігаються разом)
- Це не реляційна база даних



DATASTORE

- Що ж нам дає Datastore?
 - можливість будувати масштабовані застосування
 - можливість працювати з грандіозними масивами даних
 - реплікацію



ЗБЕРЕЖЕННЯ ДАНИХ

- З чим ви працювали і наведіть приклади:
 - local filesystem
 - shared filesystem
 - cloud file storage
 - relational databases
 - NoSQL databases
 - Google App Engine Datastore



DATASTORE

- Коли ми працюємо з Google Datastore ми маємо використовувати наступні абстракції:
 - kind
 - дуже схоже на концепт клас з ООП
 - це загальний вид сутностей
 - entity
 - конкретний запис в kind
 - property
 - властивості якими володіють entity



DATASTORE

Технологія	Структура	Екземпляр	Властивість
Datastore	Kind	Entity	Property
ООП	Class	Object	Field
RDB	Table	Row	Column



DATASTORE

- Відмінностей дуже багато, але спочатку можете уявляти саме так



DATASTORE

- Давайте спробуємо розібратися як зберігати об'єкти в Datastore
- Для цього ми розберемося з
 - Objectify
 - Entity
 - unique id
 - admin console
 - відобування entity з Datastore



ENTITY

- `import com.googlecode.objectify.annotation.Entity;`
- `import com.googlecode.objectify.annotation.Id;`
- `@Entity`
- `public class Student{`
 - `@Id Long studentId;`
 - `String name;`
 - `Integer course`
 - `//getters and setters`
- `}`

- Більш докладно про Entity
 - <https://cloud.google.com/appengine/docs/java/datastore/entities>
- Більш докладно про Objectify
 - <https://code.google.com/p/objectify-appengine/wiki/Concepts>



PROPERTY TYPES

- Як ви думаєте, що відбувається з вашим об'єктом на етапі збереження його в Datastore?
- Що відбувається з його полями?



PROPERTY TYPES

- Основні типи, що підтримуються:
 - integers
 - floating-point numbers
 - strings
 - dates
 - binary data
- Але типів набагато більше
 - [https://cloud.google.com/appengine/docs/java/datastore/entities#Java Property types and value types](https://cloud.google.com/appengine/docs/java/datastore/entities#Java_Property_types_and_value_types)



ЗБЕРЕЖЕННЯ

- Давайте повернемося до нашого проекту з конференціями
- В нас був профайл користувача, тепер ми хочемо зберігати його в datastore
- Що ми маємо зробити спочатку?



ЗБЕРЕЖЕННЯ

- @Entity
- **public class Profile {**
 - String displayName;
 - String mainEmail;
 - TeeShirtSize teeShirtSize;

 - @Id String userId;
 - ...
- }



DATASTORE KEYS

- Коли ви зберігаєте Entity в datastore, вона отримує у відповідність ключ
- Ключ генерується datastore
- Цей ключ унікально ідентифікує Entity
- Існує два способи отримати ключ від datastore



DATASTORE KEYS

□ Перший:

- В нас є проста Entity
 - Student
 - name
 - course
- коли ми будемо зберігати дану сутність, datastore поверне нам ключ



DATASTORE KEYS

□ Другий

- В нас є Entity в якій визначене унікальне поле
 - Student
 - id
 - name
 - course
- коли ми будемо зберігати дану сутність, datastore поверне нам ключ, що буде згенерований на основі даного унікального поля
- Перевага даного підходу полягає в тому, що ви можете забрати сутність з бази за цим полем id, а в першому випадку в вас немає такого поля і потрібно використовувати додаткові механізми

□ *primary key*



ЗБЕРЕЖЕННЯ

- Розберемося з збереженням даних використовуючи бібліотеку Objectify на прикладі
 - <https://code.google.com/p/objectify-appengine/wiki/BasicOperations#Saving>



ЗБЕРЕЖЕННЯ

- Для збереження Entity
 - `ObjectifyService.ofy().save().entity(entity).now();`
- Гарна практика визначити статичний метод `ofy` що буде повертати нам даний сервіс
 - подивимося `OfyService` клас нашого проекту
- Таким чином, збереження нашого профайлу буде виглядати наступним чином:
 - `ofy().save().entity(profile).now();`



ЗБЕРЕЖЕННЯ

- Внесемо необхідні зміни в проект і запустимо його
- Запустимо консоль управління
 - <http://apis-explorer.appspot.com/apis-explorer/?base=http://localhost:8080/ah/api>
- Спробуємо тепер наш метод saveProfile



fields

Selector specifying which fields to include in a partial response.

[Use fields editor](#)

Request body

```
{
  "displayName": "Andrii Glybovets"
  "teeShirtSize": "XL"
}
```

[Execute](#)

conference.saveProfile executed moments ago time to execute: 941 ms

Request

POST http://localhost:8080/_ah/api/conference/v1/profile

Content-Type: application/json

Authorization: Bearer ya29.8gCA5tQDnOnmBxSG9LyGmT8C25Mx1Ir2gVgTEAw0P9JGuu2oaX7WAifs7PBmGvPm6lCDR0s5465DAw

X-JavaScript-User-Agent: Google APIs Explorer

```
-{
  "displayName": "Andrii Glybovets",
  "teeShirtSize": "XL"
}
```

Response

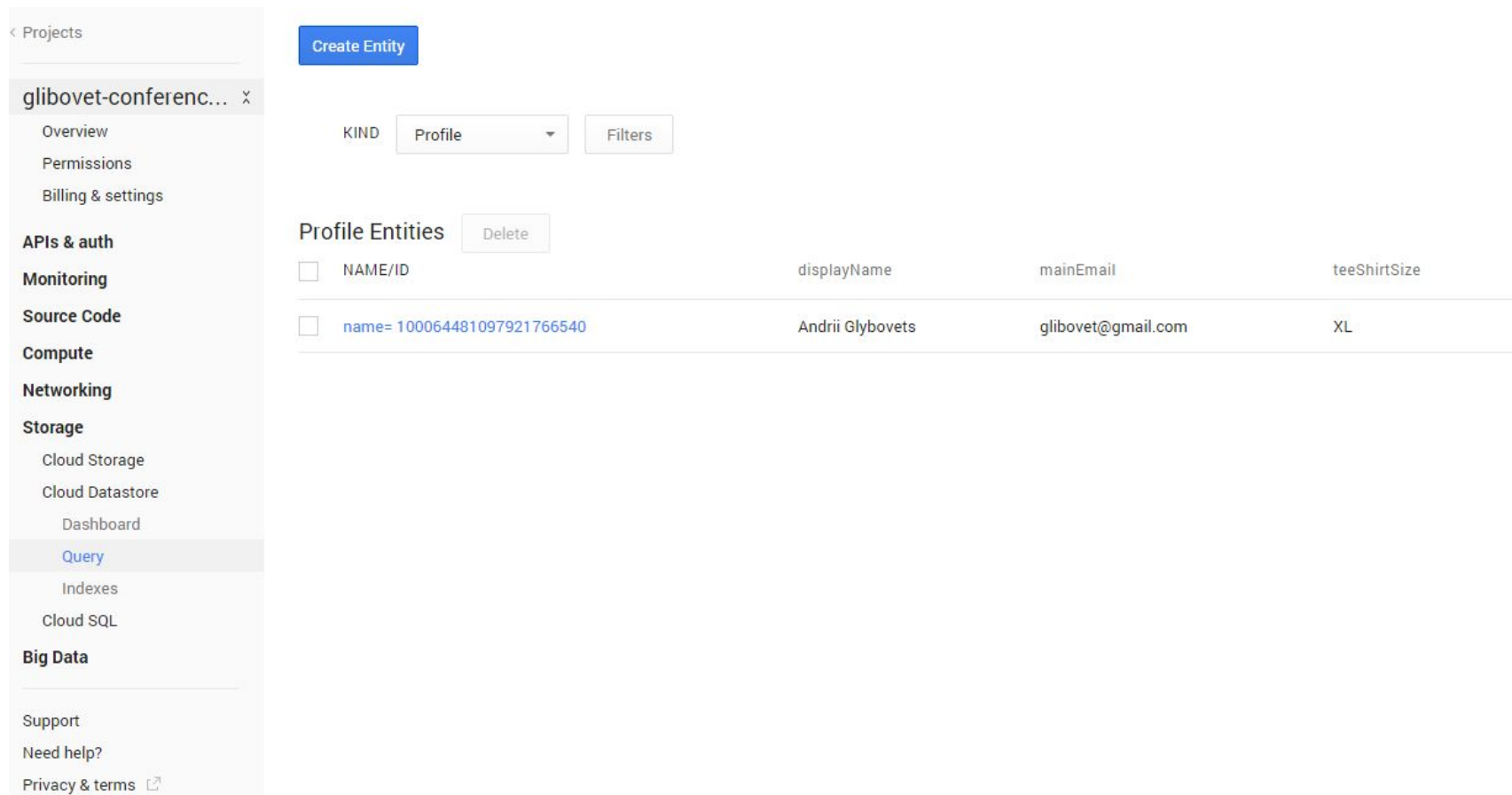
200 OK

[- Show headers -](#)

```
-{
  "displayName": "Andrii Glybovets",
  "mainEmail": "glibovet@gmail.com",
  "teeShirtSize": "XL",
  "userId": "100064481097921766540",
  "kind": "conference#resourcesItem",
  "etag": "\"gIgmveYe55JDf053jb0bI1QB1qU/X734R5fbc1LnNPt7iz7dduAcTg4\""
}
```

ЗБЕРЕЖЕННЯ

□ Тепер підемо в панель управління в datastore



The screenshot displays the Google Cloud Datastore console interface. On the left is a navigation sidebar with categories like 'Projects', 'APIs & auth', 'Monitoring', 'Source Code', 'Compute', 'Networking', 'Storage', and 'Big Data'. The 'Query' option is highlighted. The main content area shows a 'Create Entity' button, a 'KIND' dropdown set to 'Profile', and a 'Filters' button. Below this is a table titled 'Profile Entities' with a 'Delete' button. The table has columns for 'NAME/ID', 'displayName', 'mainEmail', and 'teeShirtSize'. One entity is listed with the ID 'name= 100064481097921766540', displayName 'Andrii Glybovets', mainEmail 'glibovet@gmail.com', and teeShirtSize 'XL'.

<input type="checkbox"/>	NAME/ID	displayName	mainEmail	teeShirtSize
<input type="checkbox"/>	name= 100064481097921766540	Andrii Glybovets	glibovet@gmail.com	XL

ЗБЕРЕЖЕННЯ

- Тут ви можете:
 - переглядати свої сутності
 - змінювати їх
 - робити запити до них



KEY CONFLICTS

- Ми говорили, що є два способи отримати ключ:
 - автоматичний
 - в даному випадку, Google відповідає за унікальність ключів
 - на основі нашого id
 - якщо в ваших Entity співпадуть ID, datastore згенерує однаковий ключ



ВИДОБУВАННЯ ДАНИХ

- Видобування даних таке ж просте як і збереження:
 - `Key key = Key.create(Entity.class, id);`
 - `Entity entity = ofy().load().key(key).now();`



ВИДОБУВАННЯ ДАНИХ

- Подивимося на практиці
- В нас в АРІ є метод `getProfile`
- Що необхідно зробити?
- `@ApiMethod(name = "getProfile", path = "profile", httpMethod = HttpMethod.GET)`
- `public Profile getProfile(final User user) throws UnauthorizedException {`
- `if (user == null) {`
- `throw new UnauthorizedException("Authorization required");`
- `}`
- `// TODO`
- `// load the Profile Entity`
- `String userId = ""; // TODO`
- `Key key = null; // TODO`
- `Profile profile = null; // TODO load the Profile entity`
- `return profile;`
- `}`



ВИДОБУВАННЯ ДАНИХ

- `@ApiMethod(name = "getProfile", path = "profile", httpMethod = HttpMethod.GET)`
- `public Profile getProfile(final User user) throws UnauthorizedException {`
 - `if (user == null) {`
 - `throw new UnauthorizedException("Authorization required");`
 - `}`
 - `String userId = user.getUserId();`
 - `Key key = Key.create(Profile.class,userId);`
 - `Profile profile = (Profile)ofy().load().key(key).now();`
 - `return profile;`
- `}`



ВИДОБУВАННЯ ДАНИХ

□ Або так

- `@ApiMethod(name = "getProfile", path = "profile", httpMethod = HttpMethod.GET)`
- `public Profile getProfile(final User user) throws UnauthorizedException {`
 - `if (user == null) {`
 - `throw new UnauthorizedException("Authorization required");`
 - `}`
 - `String userId = user.getUserId();`
 - `Key<Profile> key = Key.create(Profile.class,userId);`
 - `Profile profile = ofy().load().key(key).now();`
 - `return profile;`
 - `}`




ВИДОБУВАННЯ ДАНИХ

- Зробимо зміни та запустимо проект
- Протестуємо наш API
- Збережіть профайл
- потім спробуйте `getProfile`



ВИДОБУВАННЯ ДАНИХ

Services > conference API v1 > conference.getProfile

Authorize requests using OAuth 2.0: ON 

fields

Selector specifying which fields to include in a partial response.

[Use fields editor](#)

Execute

conference.getProfile executed moments ago time to execute: 294 ms

Request

```
GET http://localhost:8080/_ah/api/conference/v1/profile
```

```
Authorization: Bearer ya29.8gCA5tQDnOnmBxSG9LyGmT8C25Mx1Ir2gVgTEAw0P9JGuu2oaX7WAiFs7PBmGvPm6lCDR0s5465DAw
X-JavaScript-User-Agent: Google APIs Explorer
```

Response

```
200 OK
```

- Show headers -

```
-{
  "displayName": "Glybovets",
  "mainEmail": "glibovet@gmail.com",
  "teeShirtSize": "L",
  "userId": "100064481097921766540",
  "kind": "conference#resourcesItem",
  "etag": "\"\gIgmveYe55JDF053jbObilQBiqU/j84E9FAHpolz21aElVsnJYknLbs\""
}
```



Оновлення ENTITY

- Якщо ви звернете увагу на метод `saveProfile`
- Ми кожний раз створюємо новий `Profile`
- Чому ми це робимо?
- Яка проблема з ключами?



Оновлення ENTITY

- Ми кожен раз створюємо новий об'єкт і затираємо старий
- Інакше ми б отримали конфлікт ключів
- Іноколи нам хочеться просто оновити дані існуючої сутності
- Як це зробити?



Оновлення ENTITY

- Мені потрібно лише трохи змінити метод `saveProfile`
 - `Profile profile = getProfile(user);`
 - **`if (profile == null)`**
 - `profile = new Profile(userId, displayName, mainEmail, teeShirtSize);`
 - **`else`**
 - `profile.update(displayName, teeShirtSize);`
- та дописати метод `update` в `Profile` клас



До змін

conference.getProfile executed moments ago time to execute: 2192 ms

Request

```
GET http://localhost:8080/_ah/api/conference/v1/profile
```

```
Authorization: Bearer ya29.8gBEptnsYxcGd_r-zhEdXJ6-LA1_UtmIBi_RhG8XSFVnRTxaG2r3CyCXVm7gbz0j1lnVkSMV9GZr9w  
X-JavaScript-User-Agent: Google APIs Explorer
```

Response

200 OK

- Show headers -

```
-{  
  "displayName": "Glybovets",  
  "mainEmail": "glibovet@gmail.com",  
  "teeShirtSize": "L",  
  "userId": "100064481097921766540",  
  "kind": "conference#resourcesItem",  
  "etag": "\"gIgmveYe55JDf053jb0bIlQBIqU/j84E9FAHpolz21aElVsnJYknLbs\""  
}
```



До змін

- ▣ Я змінив лише одне поле
- ▣ Що відбулося?

conference.saveProfile executed moments ago time to execute: 587 ms

Request

POST http://localhost:8080/_ah/api/conference/v1/profile

Content-Type: application/json

Authorization: Bearer ya29.8gBEptnsYxcGd_r-zhEdXJ6-LA1_UtmIBi_RhG8XSFVnRTxaG2r3CyCXVm7gbz0j1lnVkSMV9GZr9w

X-JavaScript-User-Agent: Google APIs Explorer

```
-{
  "teeShirtSize": "XL"
}
```

Response

200 OK

- Show headers -

```
-{
  "displayName": "glibovet",
  "mainEmail": "glibovet@gmail.com",
  "teeShirtSize": "XL",
  "userId": "100064481097921766540",
  "kind": "conference#resourcesItem",
  "etag": "\"gIgmveYe55JDf053jb0bIlQBIIQU/RLNcUcBLMfvGQB8U-jrChHwa2Ug\""
}
```



Після змін

conference.getProfile executed moments ago time to execute: 256 ms

Request

```
GET http://localhost:8080/_ah/api/conference/v1/profile
```

```
Authorization: Bearer ya29.8gBEptnsYxcGd_r-zhEdXJ6-LA1_UtmIBi_RhG8XSFVnRTxaG2r3CyCXVm7gbz0j1lnVkSMV9GZr9w  
X-JavaScript-User-Agent: Google APIs Explorer
```

Response

200 OK

- Show headers -

```
-{  
  "displayName": "Glybovets",  
  "mainEmail": "glibovet@gmail.com",  
  "teeShirtSize": "L",  
  "userId": "100064481097921766540",  
  "kind": "conference#resourcesItem",  
  "etag": "\"gIgmveYe55JDfO53jb0bIlQBiqU/j84E9FAHpolz21aElVsnJYknLbs\""  
}
```



Після змін

conference.saveProfile executed moments ago time to execute: 1519 ms

Request

```
POST http://localhost:8080/_ah/api/conference/v1/profile
```

```
Content-Type: application/json
```

```
Authorization: Bearer ya29.8gBEptnsYxcGd_r-zhEdXJ6-LA1_UtmIBi_RhG8XSFVnRTxaG2r3CyCXVm7gbz0j11nVkSMV9GZr9w
```

```
X-JavaScript-User-Agent: Google APIs Explorer
```

```
-{  
  "teeShirtSize": "XL"  
}
```

Response

200 OK

- Show headers -

```
-{  
  "displayName": "Glybovets",  
  "mainEmail": "glibovet@gmail.com",  
  "teeShirtSize": "XL",  
  "userId": "100064481097921766540",  
  "kind": "conference#resourcesItem",  
  "etag": "\"gIgmveYe55JDfO53jb0bIlQBIqU/03li8EoFodippIKmYai6s9mQ77M\""  
}
```

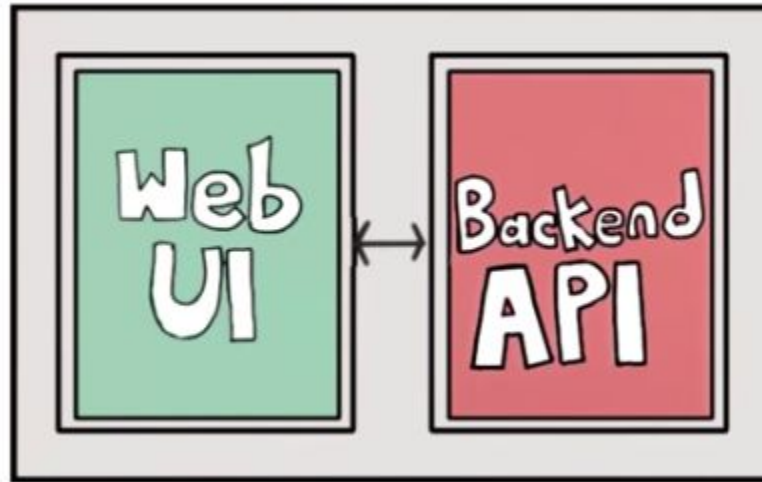


CONFERENCE CENTRAL

- Ми з вами зробили два методи з API
- В проекті який ми імпортували наявний повний Web UI
- Таким чином як тільки ми зробили API по роботі з профайлом користувача, він почав працювати в Веб інтерфейсі
- Давайте подивимося в controllers.js на save and get profile

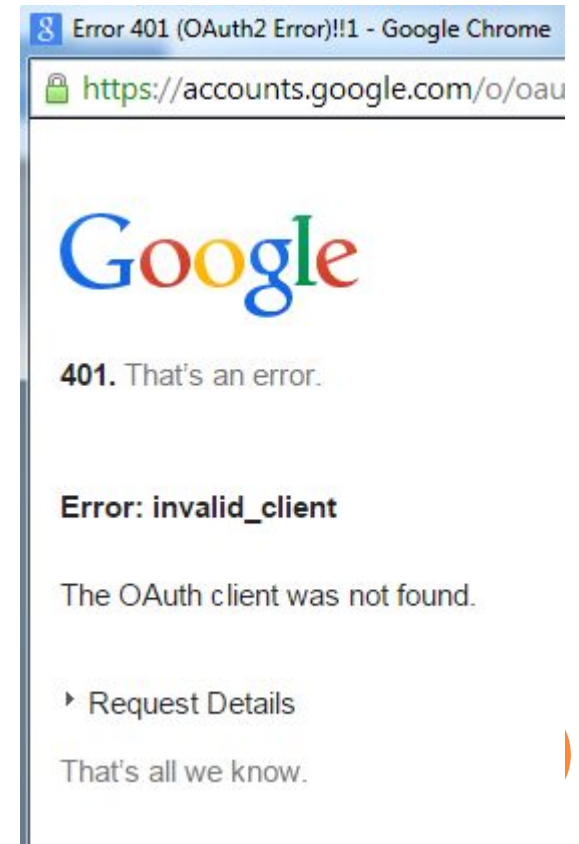


WEB UI AND BACKEND API



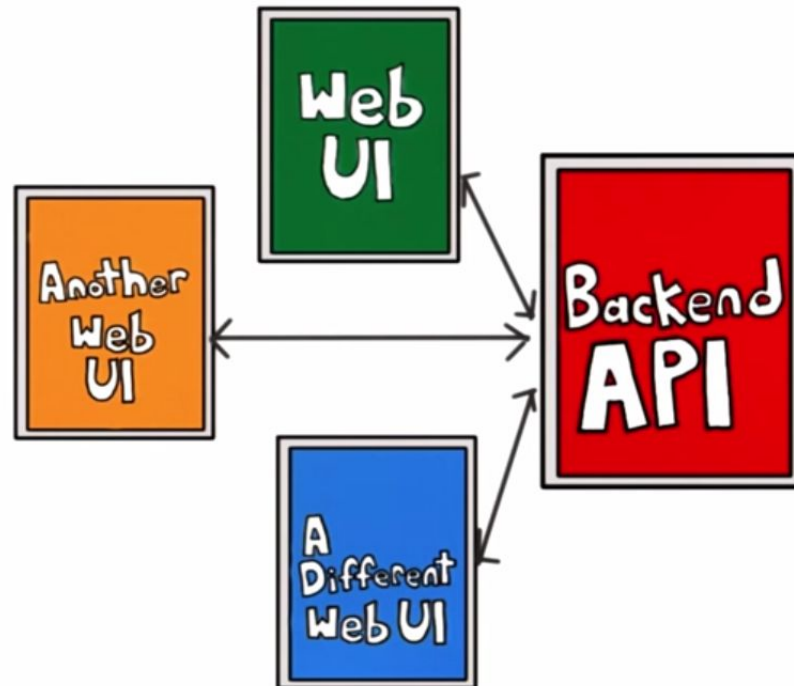
WEB UI AND BACKEND API

- Якщо ви зробили все про що ми говорили і запустили застосування, то ви мали б отримати помилку
- Це тому, що нам потрібно авторизувати Web UI в Backend API



WEB UI AND BACKEND API

- В нашому випадку обидві ці частини знаходяться в одному проекті, але це не обов'язково!!!



WEB UI AND BACKEND API

- В будь-якому випадку (один проект, окремі проекти) ви маєте авторизувати ваше застосування для використання API



WEB UI AND BACKEND API

- Подивимося на ConferenceApi
- @Api(name = "conference",
 - version = "v1",
 - scopes = { Constants.*EMAIL_SCOPE* },
 - clientIds = {
 - Constants.*WEB_CLIENT_ID*,
 - Constants.*API_EXPLORER_CLIENT_ID* },
 - description = "API for the Conference Central Backend application.")
- **public class ConferenceApi {**



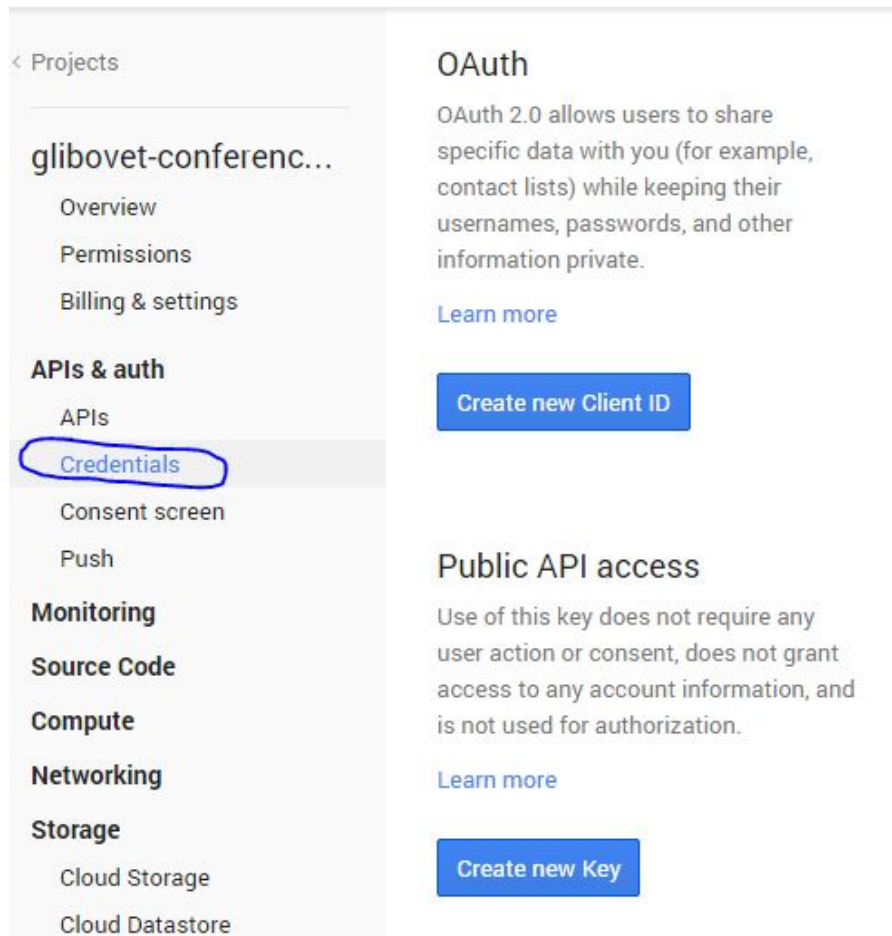
WEB UI AND BACKEND API

- Подивимося в Constants
 - **public static final String *WEB_CLIENT_ID* = *"replace this with your Web client ID"*;**
- Ми маємо замінити цю стрічку нашим ID
- Ви можете знайти його в вашій девелоперській консолі на App Engine



WEB UI AND BACKEND API

Google Developers Console



The screenshot shows the Google Developers Console interface. On the left is a navigation sidebar with a 'Projects' breadcrumb and a list of menu items: 'glibovet-conferenc...', 'Overview', 'Permissions', 'Billing & settings', 'APIs & auth', 'APIs', 'Credentials' (highlighted with a blue circle), 'Consent screen', 'Push', 'Monitoring', 'Source Code', 'Compute', 'Networking', and 'Storage' (with sub-items 'Cloud Storage' and 'Cloud Datastore'). The main content area is titled 'OAuth' and contains a paragraph explaining OAuth 2.0, a 'Learn more' link, and a blue button labeled 'Create new Client ID'. Below this, there is a section titled 'Public API access' with a paragraph explaining its use, another 'Learn more' link, and a blue button labeled 'Create new Key'. An orange circle is visible in the bottom right corner of the page.

< Projects

glibovet-conferenc...

- Overview
- Permissions
- Billing & settings

APIs & auth

- APIs
- Credentials**
- Consent screen
- Push

Monitoring

Source Code

Compute

Networking

Storage

- Cloud Storage
- Cloud Datastore

OAuth

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.

[Learn more](#)

[Create new Client ID](#)

Public API access

Use of this key does not require any user action or consent, does not grant access to any account information, and is not used for authorization.

[Learn more](#)

[Create new Key](#)

WEB UI AND BACKEND API

OAuth

OAuth 2.0 allows users to share specific data with you (like contact lists) while keeping their usernames, passwords and other information private.

[Learn more](#)

[Create new Client ID](#)

Public API access


Use of this key does not require user action or consent, and it does not have access to any account information. This key is not used for authorization.

[Learn more](#)

Create Client ID

APPLICATION TYPE

- Web application**
Accessed by web browsers over a network.
- Service account**
Calls Google APIs on behalf of your application instead of an end-user.
[Learn more](#)
- Installed application**
Runs on a desktop computer or handheld device (like Android or iPhone).

 To create a Web Client ID or an Installed Application Client, you need to set a product name in the consent screen.

[Configure consent screen](#)

[Cancel](#)



WEB UI AND BACKEND API

OAuth

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.

[Learn more](#)

Create new Client ID

Client ID for web application

CLIENT ID	508913531626-2bi19p760ej2h5fjc1g3c6himhkjbpcu.apps.googleusercontent.com
EMAIL ADDRESS	508913531626-2bi19p760ej2h5fjc1g3c6himhkjbpcu@developer.gserviceaccount.com
CLIENT SECRET	USa0ThiiOa9hm0CfZ_AolqLY
REDIRECT URIS	https://www.example.com/oauth2callback
JAVASCRIPT ORIGINS	https://www.example.com

Edit settings

Reset secret

Download JSON

Delete



WEB UI AND BACKEND API

Client ID for web application

CLIENT ID	508913531626-2bi19p760ej2h5fjc1g3c6himhkjbpcu.apps.googleusercontent.com
EMAIL ADDRESS	508913531626-2bi19p760ej2h5fjc1g3c6himhkjbpcu@developer.gserviceaccount.com
CLIENT SECRET	USa0ThiiOa9hm0CfZ_AolqLY
REDIRECT URIS	none
JAVASCRIPT ORIGINS	http://localhost:8080 https://glibovet-conference-engine.appspot.com/

[Edit settings](#)[Reset secret](#)[Download JSON](#)[Delete](#)

WEB UI AND BACKEND API

- Тепер в нас є client ID
- Вставимо його в наші константи
 - **public static final String *WEB_CLIENT_ID* =
"508913531626-2bi19p760ej2h5fjc1g3c6himhkjbp
cu.apps.googleusercontent.com";**



WEB UI AND BACKEND API

- Також нам треба, щоб наше застосування вірно авторизувалося
- Для цього в нас є файл `app.js`
 - в ньому треба знайти `CLIENT_ID`
 - і також вставити наш код



□ Дякую за увагу

