Rest

Tutti lo nominano ma
pochi lo conoscono

Rest

- **REPRESENTATIONAL STATE TRANSFER**
- **È UN PARADIGMA**
- **NON È UN PROTOCOLLO!**

- **NASCE GRAZIE ALLA TESI DI Roy Fielding del 2000**

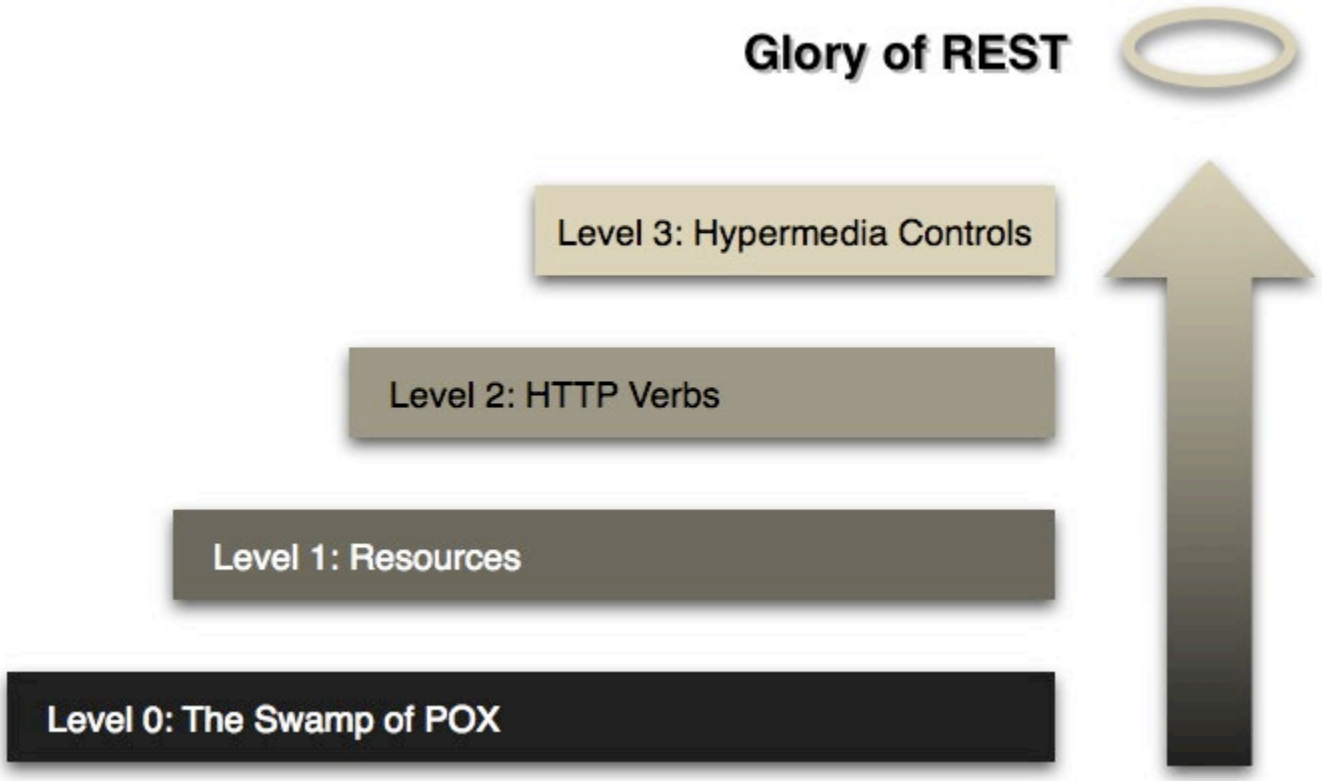
- https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf (**PAGINA 75)**

Principi del Rest

- REST is a **client-server** architecture
- REST is **stateless**
- REST is **cacheable**
- REST provides a **uniform interface** between components
- REST is a **layered system**
- REST optionally provides **code on demand**

Principi del Rest

- Richardson Maturity Model di **Martin Flower**
- https://martinfowler.com/articles/richardsonMaturityModel.html
- https://blog.restcase.com/4-maturity-levels-of-rest-api-design/



Esempio di rest

Risorsa	GET read	POST create	PUT update	DELETE
/books	Ritorna una lista di libri	Crea un nuovo libro	Aggiorna i dati di tutti i libri	Elimina tutti i libri
/books/145	Ritorna uno specifico libro	metodo non consentito (405)	Aggiorna uno specifico libro	Elimina uno specifico libro

```
GET /books/411/authors/ Restituisce la lista degli autori del libro 411
GET /books/411/authors/1 Restituisce l'autore #1 del libro 411
```

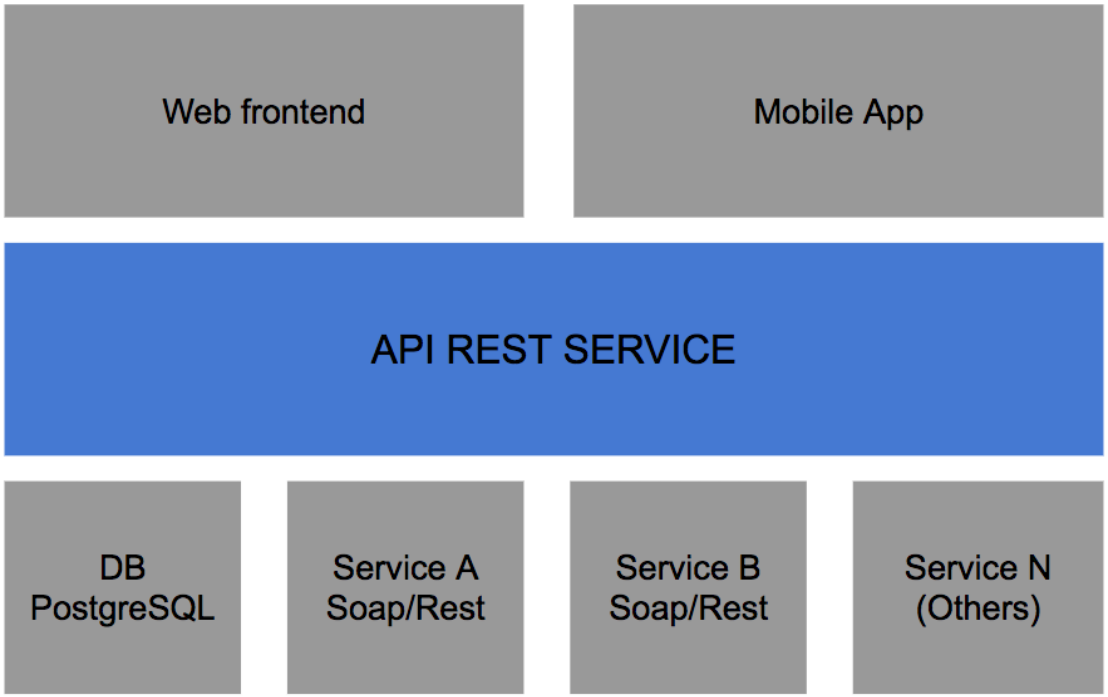
https://developers.spreaker.com/api/

https://developer.twitter.com/en/docs/api-reference-index

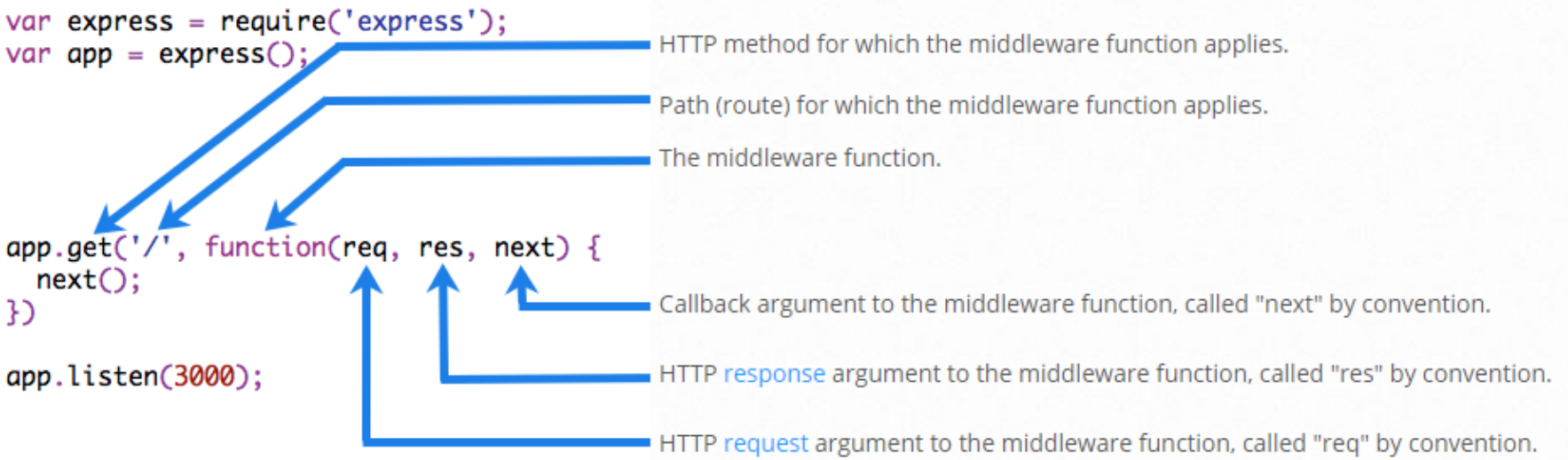
https://api.nasa.gov/

Perché tutti vogliono REST?

- **Facile**
- **Comprensibile**
- **Flessibile**



NodeJs – ExpressJS – API Rest



https://dev.to/lenmorld/quick-rest-api-with-node-and-express-in-5-minutes-336j

https://github.com/gothinkster/node-express-realworld-example-app

Security

XSS

Cos’è:

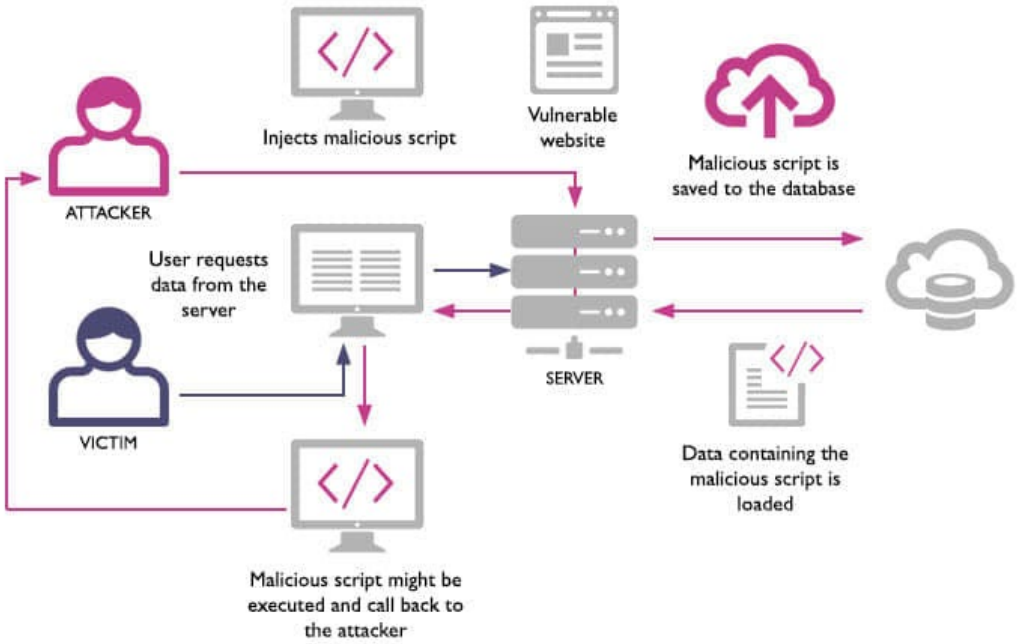
XSS: Cross-site scripting

Il **cross-site scripting** (**XSS**) è una vulnerabilità che affligge siti web dinamici che impiegano un insufficiente controllo dell'input nei form.

https://it.wikipedia.org/wiki/Cross-site_scripting

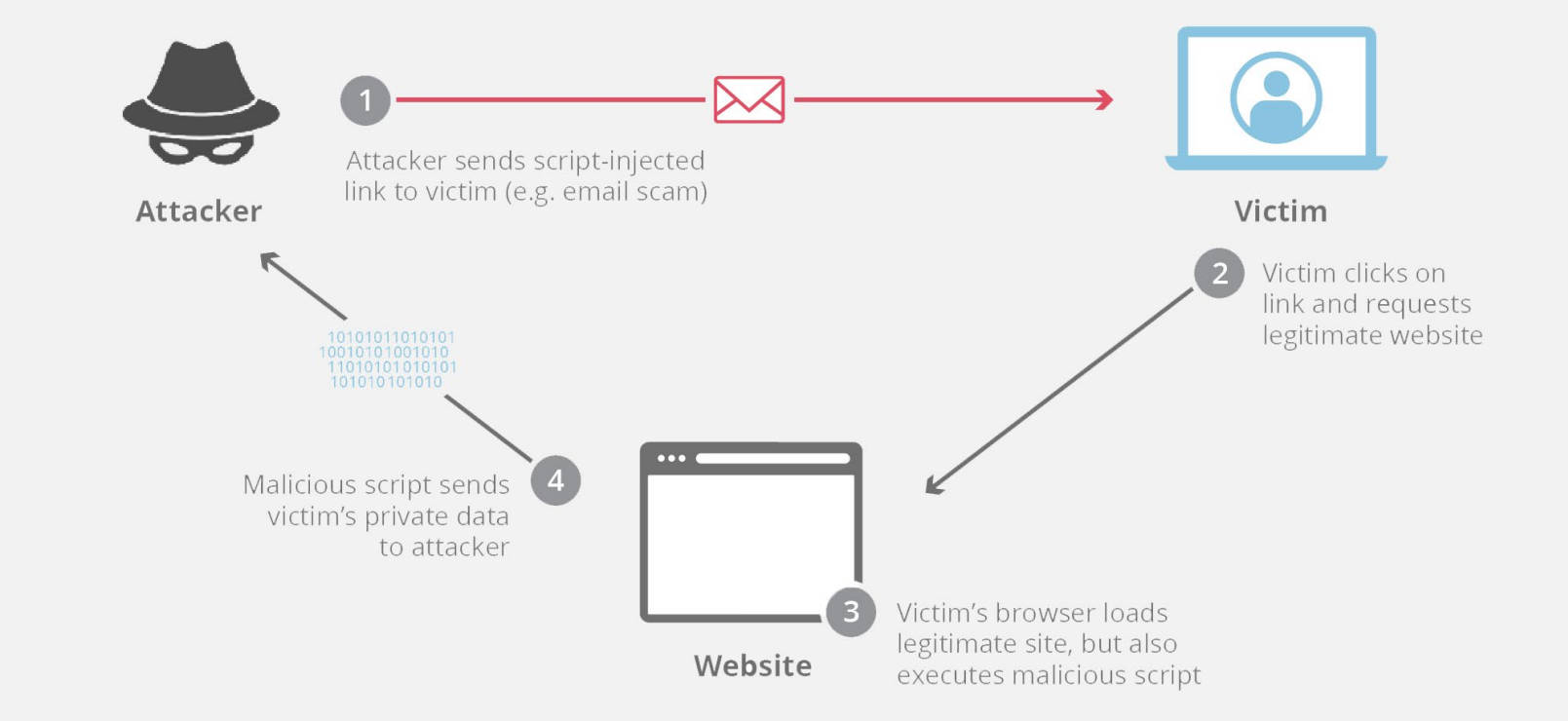
Demo:

XSS storicizzato su DB



Demo:

XSS solo su client (phishing)



Tipologie

Ogni app è un mix di tecnologie differenti

Tipologie

Nativa

Si basa su ambienti di sviluppo e SDK proprietari della piattaforma ed il codice non risulta portabile.

Ibrida

Si basa su ambienti di sviluppo e SDK scelti dallo sviluppatore ed il codice risulta facilmente portabile.

Web (PWA)

Si basa su ambienti di sviluppo e SDK web ed il codice è unico.

Tipologie

Nativa

Pro:

- Performante
- Accesso all'hardware
- GUI specifica
- Presente negli store
- API subito disponibili

Contro:

- Onerosa (per ogni piattaforma ho un SDK)
- Codice non portabile

Ibrida

Pro:

- Sviluppo veloce
- Abbastanza Performante
- GUI specifica in alcuni casi
- Codice quasi portabile
- Presente negli store

Contro:

- Accesso all'hardware limitato
- API non sempre disponibile

Web (PWA)

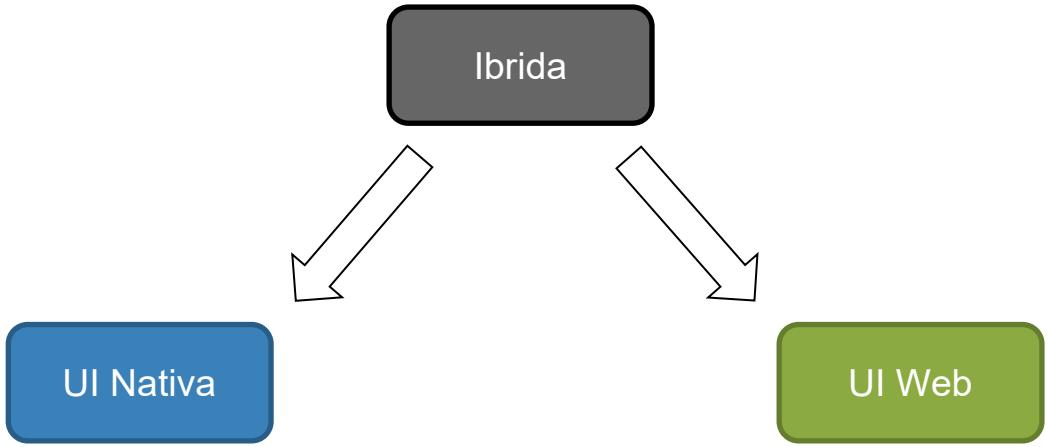
Pro:

- Sviluppo velocissimo
- Un solo codice

Contro:

- Poco performante
- GUI generica
- Accesso all'hardware limitato
- Non presente negli store

Tipologie ibride



Pro:

- Performante
- GUI specifica

Contro:

- Sviluppo lento

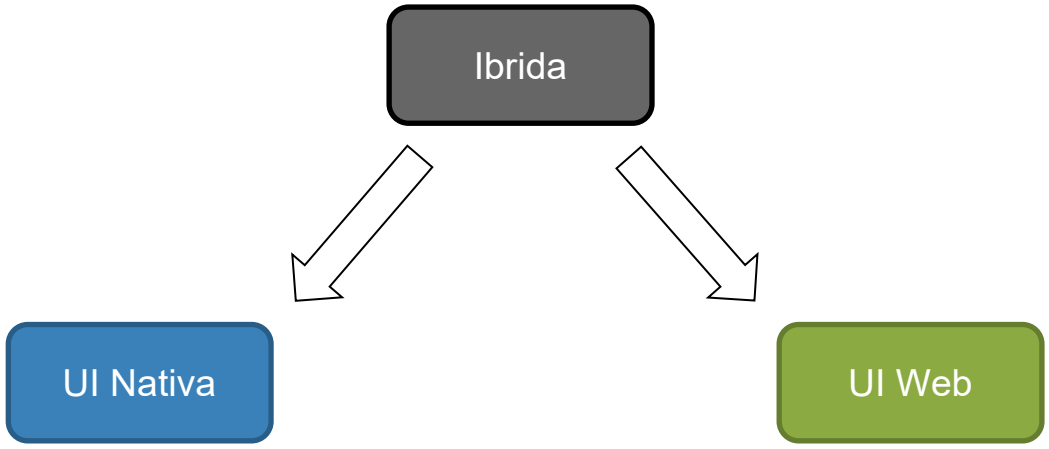
Pro:

- Sviluppo veloce

Contro:

- GUI non personalizzabile

Framework per sviluppo ibrido



- Xamarin (C#)
- React Native (JS)
- NativeScript (TS)
- MAUI (C#)
- Futter (DART)

- Ionic (TS)
- PhoneGap (JS)
- Onsen UI (JS)