WEB Platform - Uniform Interface

CS2B01 - Desarrollo Basado en Plataformas - Unidad 2

Dr. Jesus Bellido
jbellido@utec.edu.pe

UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Logros

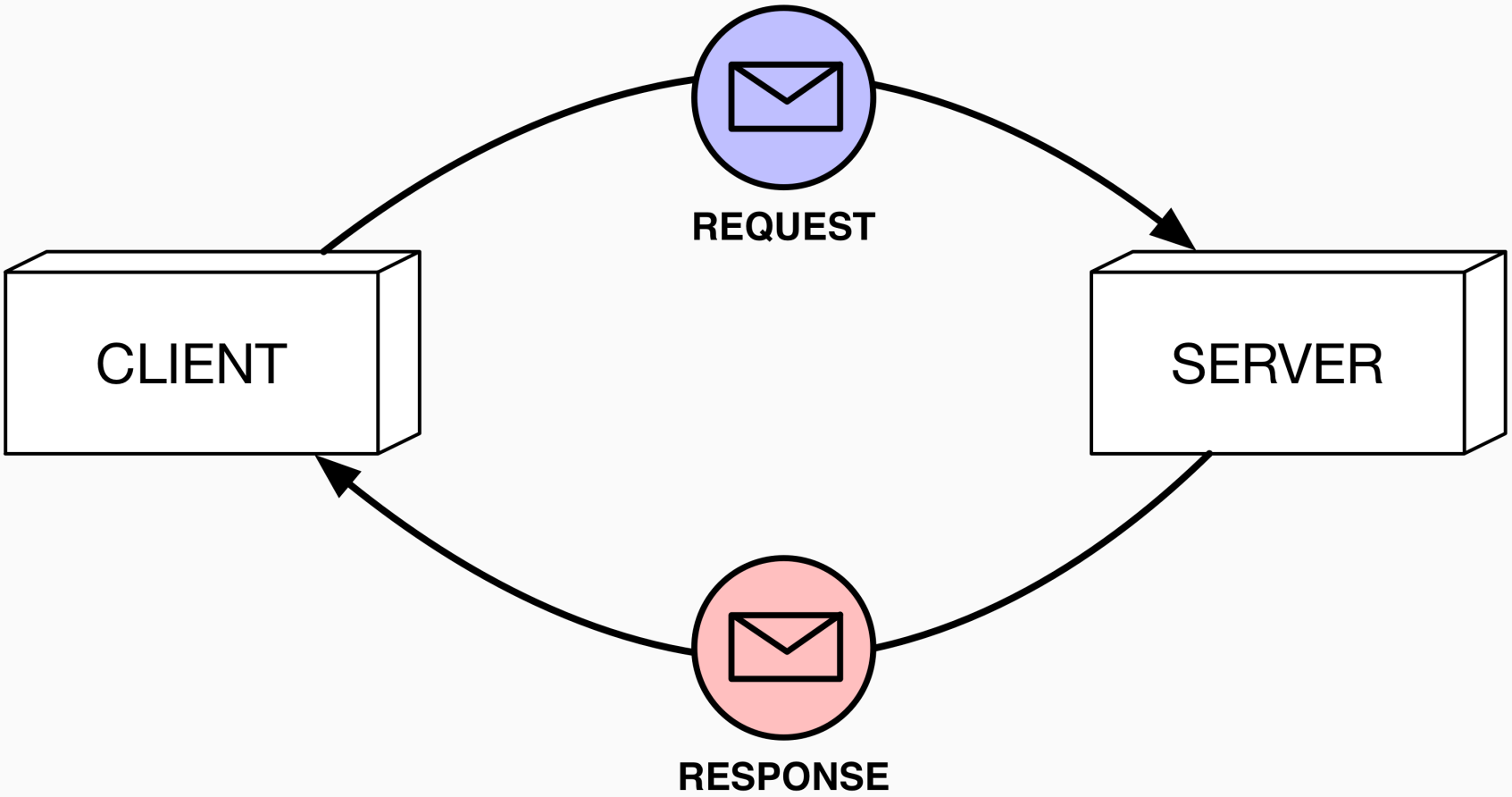
Logro de esta Sesión

Al finalizar esta unidad usted estará en la capacidad de:

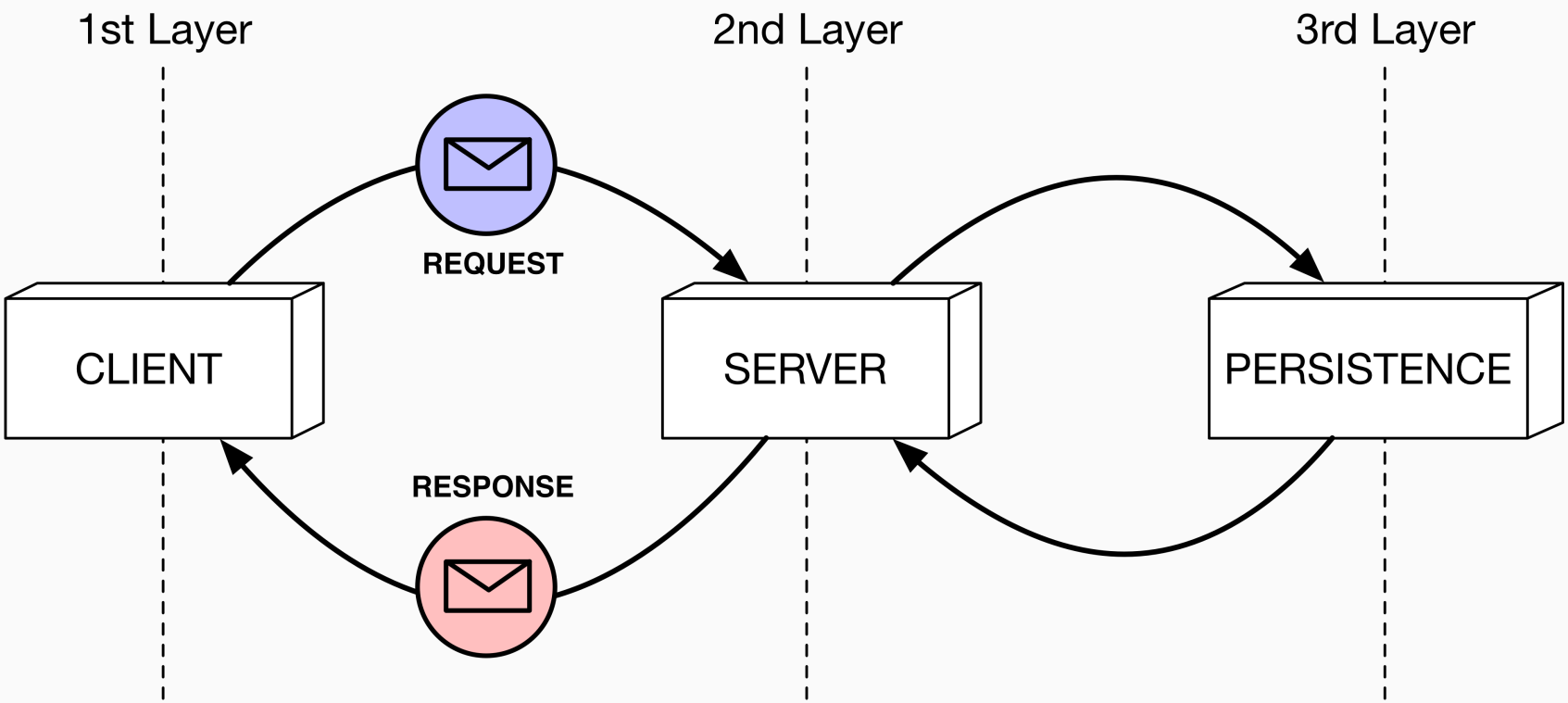
- Diseñar e implementar una aplicación web sencilla.
- Describir las restricciones que la web pone a los desarrolladores.
- Comparar y contrastar la programación web con la programación de propósito general.
- Describir las diferencias entre software como un servicio y productos de software tradicionales.
- Discutir cómo los estándares de web impactan el desarrollo de software.
- Revisar una aplicación web existente con un estándar web actual.

Introducción

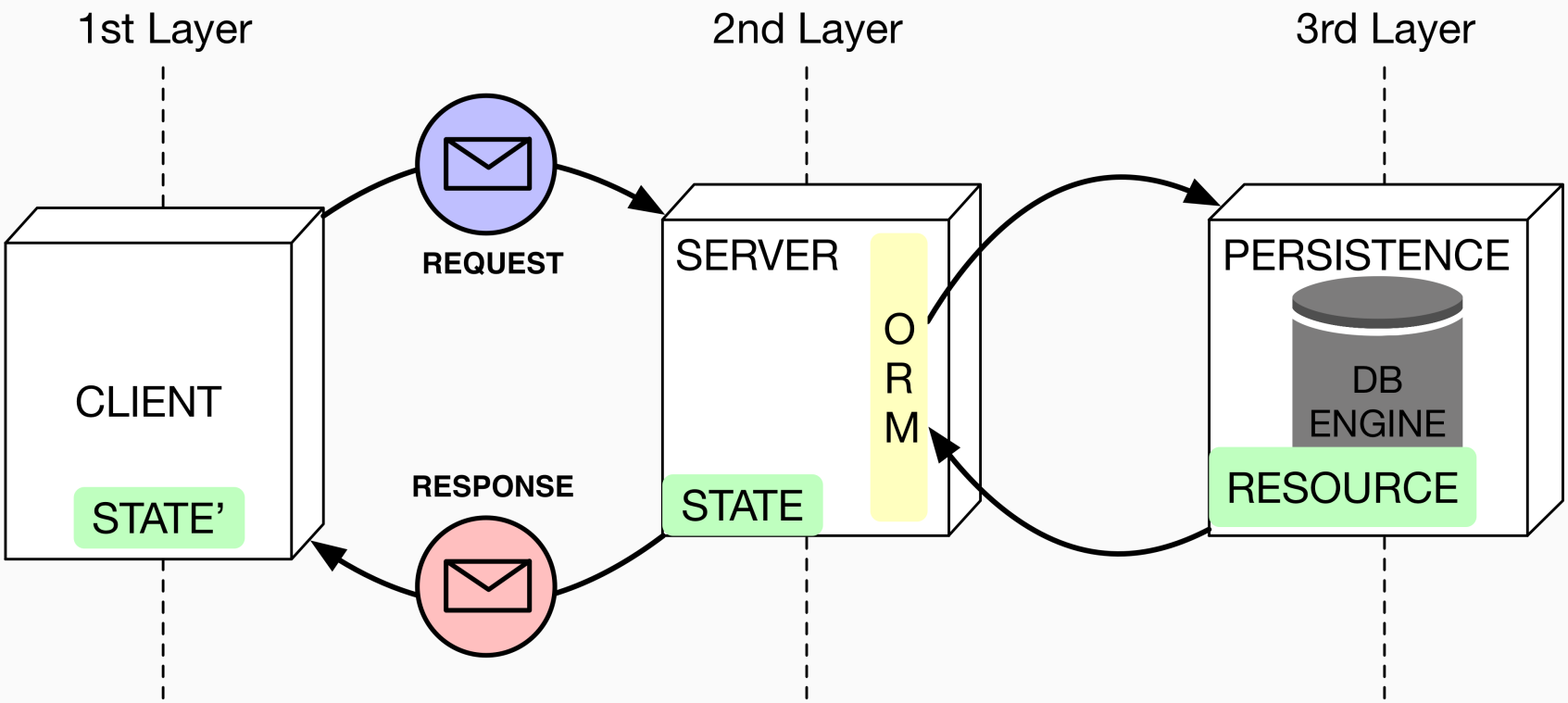
Client-Server



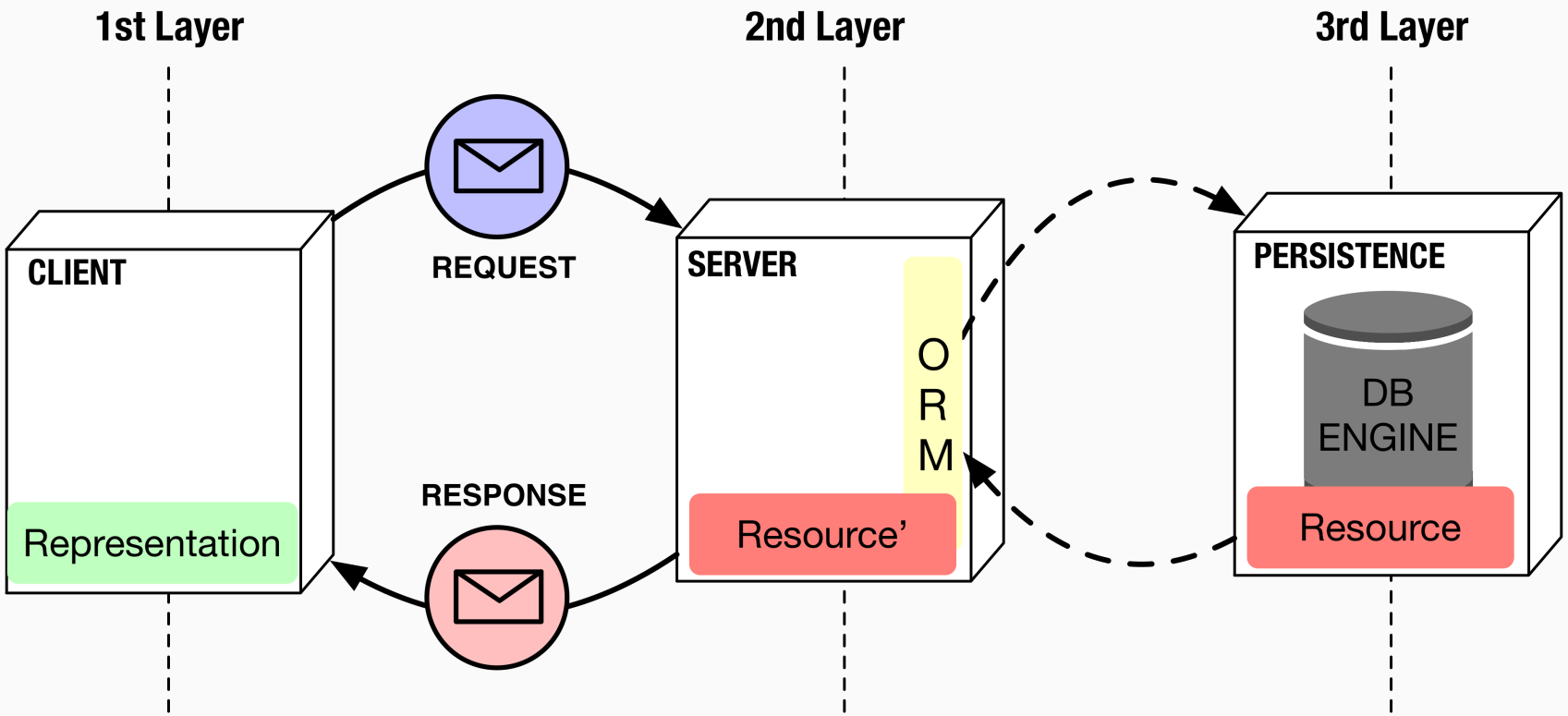
Layered System



Stateless/Stateful interaction

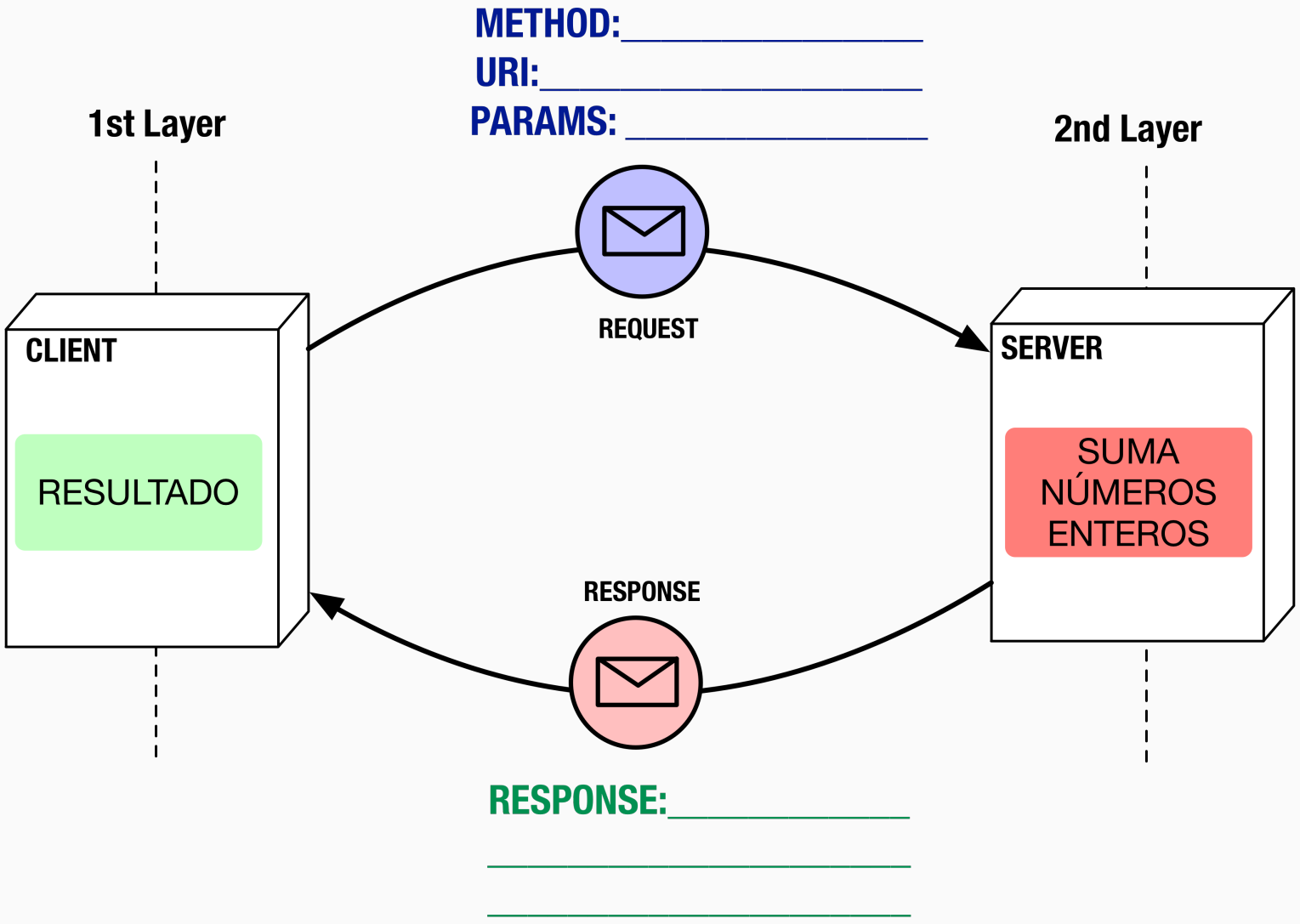


Cache in 2nd Layer

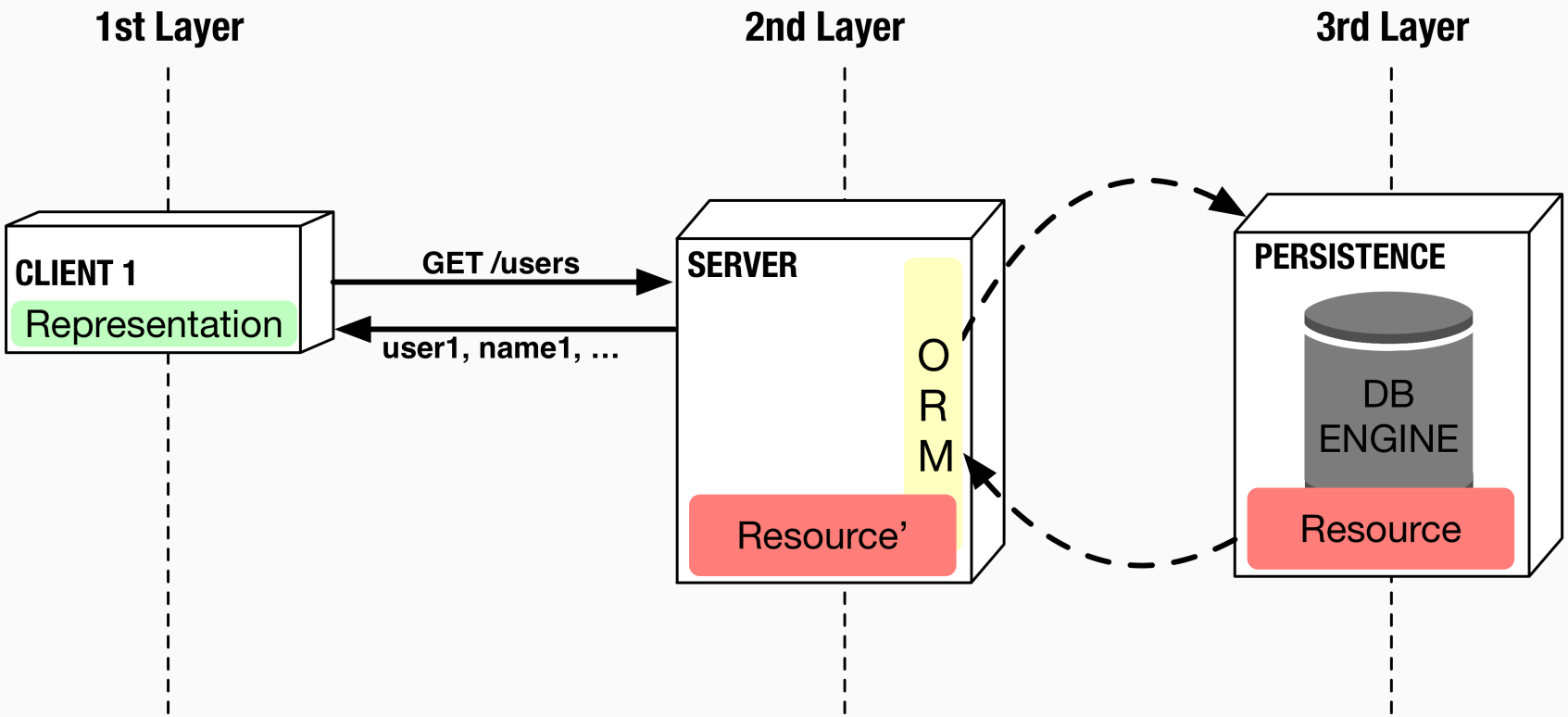


Group Work

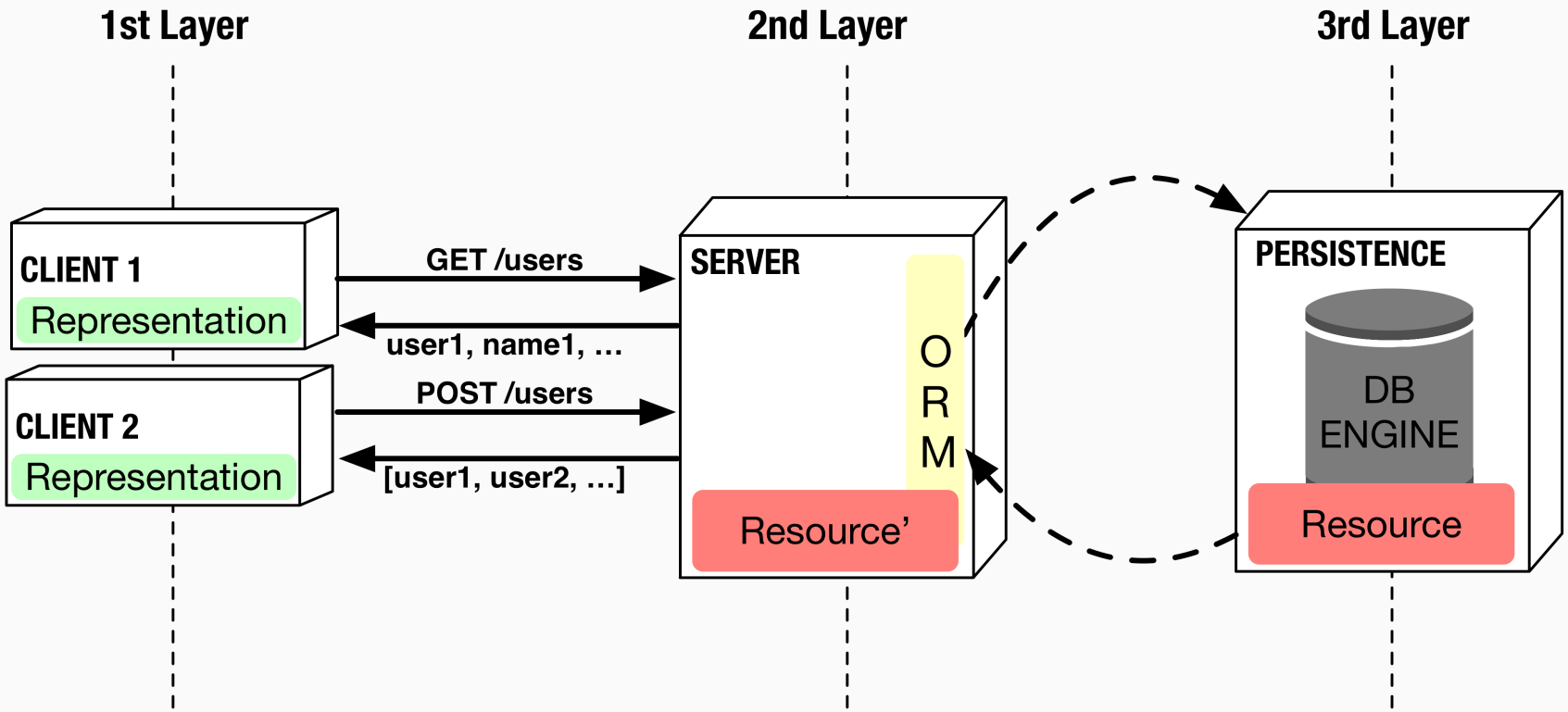
- Completa el siguiente diagrama.



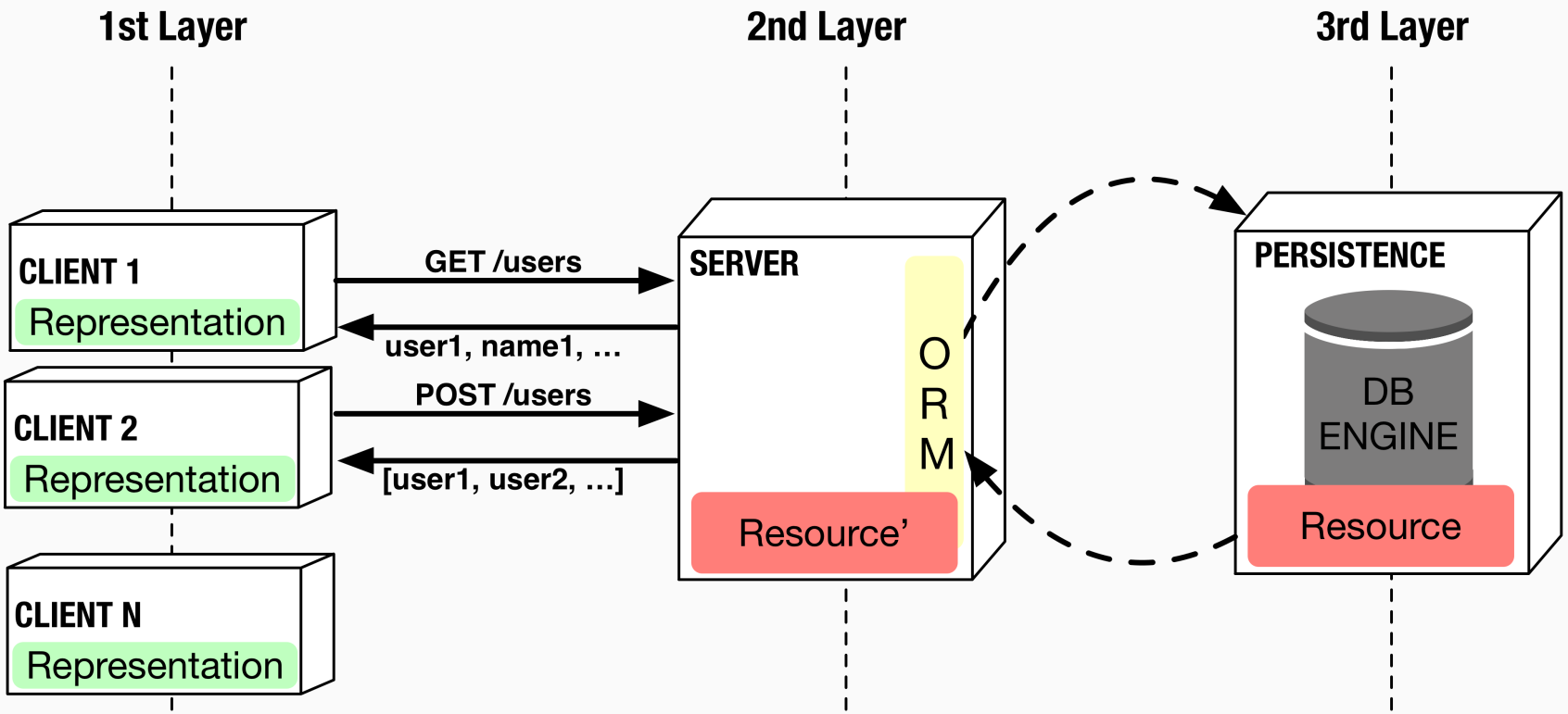
But 1 client ...


1st Layer
2nd Layer
3rd Layer
CLIENT 1
Representation
GET /users
user1, name1, ...
SERVER
O
R
M
Resource'
PERSISTENCE
DB
ENGINE
Resource

But 2 clients …



But N clients



- N clients may request user information in N different ways.
- N clients may require user representation in N different ways

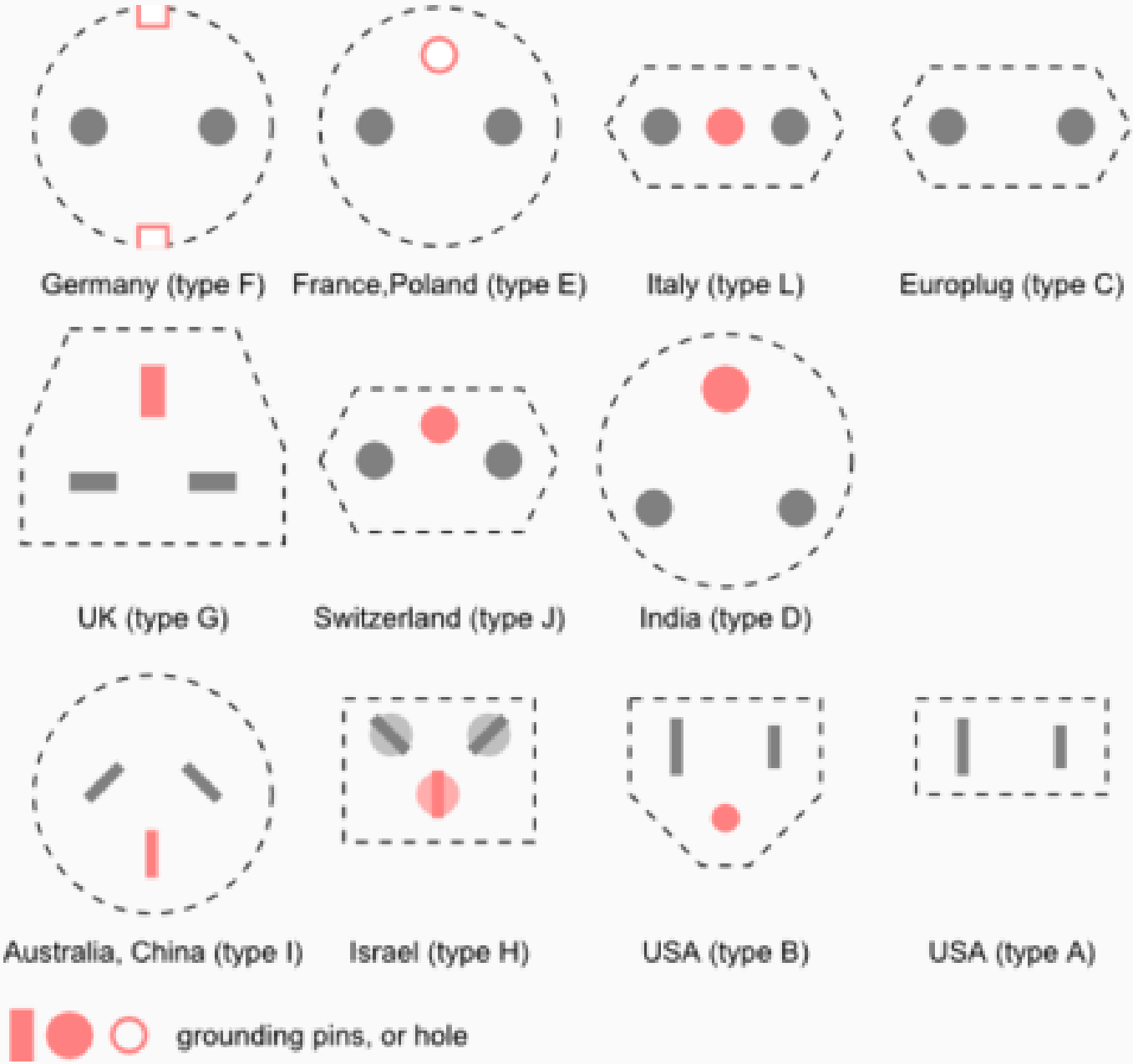
Uniform Interface

Uniform Interface

1957

2018

Uniform Interface



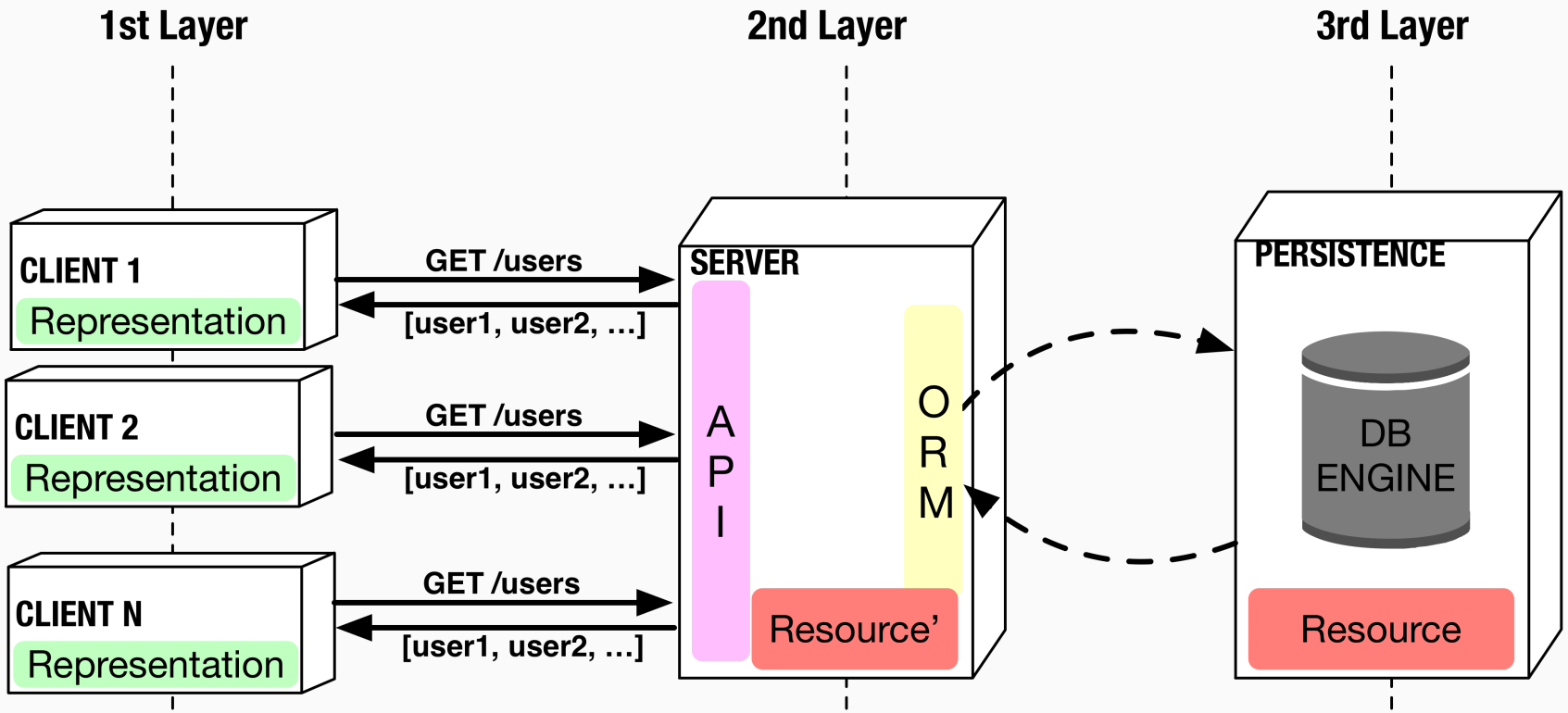
Uniform Interface

- The Uniform Interface refers to a standard way to do something regardless of what the specific activity is.
- Software Engineering principle of generality applied to interface component of a component in distributed system.
- A consistent, predictable way to query data and exercise actions regardless of the application domain

Uniform Interface

URI	Method	RQ Body	Result
`/users`	`GET`	empty	returns all users
`/users`	`POST`	user	new user created
`/users/:id`	`GET`	empty	returns single user
`/users/:id`	`PUT`	user	user updated
`/users/:id`	`DELETE`	empty	user deleted

Uniform Interface



Get Users

```
@app.route('/getUsers')
def get_user():
    key = 'getUsers'
    if key not in cache.keys():
        session = db.getSession(engine)
        dbResponse = session.query(entities.User)
        cache[key] = dbResponse;
        print("From DB")
    else:
        print("From Cache")

    users = cache[key];
    response = ""
    for user in users:
        response += user.name+";"+user.fullname
    return response
```

Get Users

```
@app.route('/getUsers')
def get_user():
    key = 'getUsers'
    if key not in cache.keys():
        session = db.getSession(engine)
        dbResponse = session.query(entities.User)
        cache[key] = dbResponse;
        print("From DB")
    else:
        print("From Cache")

    users = cache[key];
    response = ""
    for user in users:
        response += user.name+";"+user.fullname
    return response
```

User API: returns all users

```
@app.route('/users', methods = ['GET'])
def get_user():
    key = 'getUsers'
    if key not in cache.keys():
        session = db.getSession(engine)
        dbResponse = session.query(entities.User)
        cache[key] = dbResponse;
        print("From DB")
    else:
        print("From Cache")

    users = cache[key];
    response = []
    for user in users:
        response.append(user)
    return json.dumps(response, cls=connector.
       AlchemyEncoder)
```

User API: returns single user

```
@app.route('/users/<id>', methods = ['GET'])
def get_user(id):
    session = db.getSession(engine)
    users = session.query(entities.User).filter(
        entities.User.id == id)
    for user in users:
        js = json.dumps(user, cls=connector.
            AlchemyEncoder)
        return  Response(js, status=200, mimetype='
            application/json')

    message = { "status": 404, "message": "Not
        Found"}
    return Response(message, status=404, mimetype='
        application/json')
```

User API: create user

```
@app.route('/users', methods = ['POST'])
def create_user():
    c = request.get_json(silent=True)
    print(c)
    user = entities.User(
        id=c['id'],
        name=c['name'],
        fullname=c['fullname'],
        password=c['password']
    )
    session = db.getSession(engine)
    session.add(user)
    session.commit()
    return 'Created users'
```

User API: delete user

```
1 @app.route('/users/<id>', methods = ['DELETE'])
2 def remove_user(id):
3     session = db.getSession(engine)
4     users = session.query(entities.User).filter(
          entities.User.id == id)
5     for user in users:
6         session.delete(user)
7     session.commit()
8     return "DELETED"
```

User API: delete user

```
class AlchemyEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj.__class__,
            DeclarativeMeta):
             # an SQLAlchemy class
             fields = {}
             for field in [x for x in dir(obj) if
                not x.startswith('_') and x != '
                metadata']:
                 data = obj.__getattribute__(field)
                 try:
                     json.dumps(data)
                     fields[field] = data
                 except TypeError:
                     fields[field] = None
             return fields
        return json.JSONEncoder.default(self, obj)
```

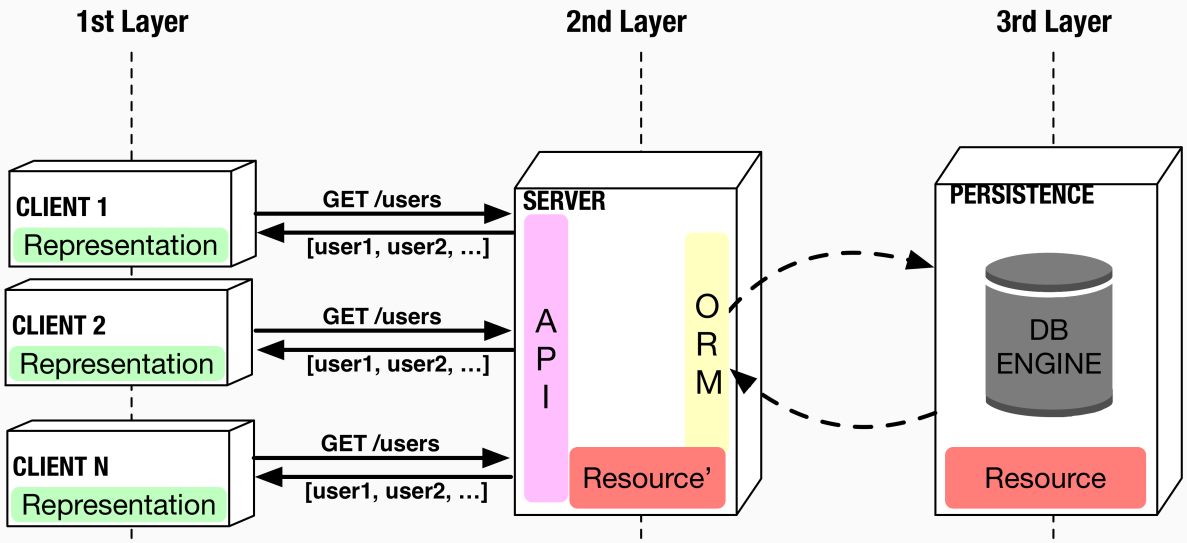
Group Work

- Implementa una API para poder administrar los mensajes del chat.

URI	Method	RQ Body	Result
`/messages`	`GET`	empty	returns all messages
`/messages`	`POST`	message	new message created
`/messages/:id`	`GET`	empty	returns single message
`/messages/:id`	`PUT`	message	message updated
`/messages/:id`	`DELETE`	empty	message deleted

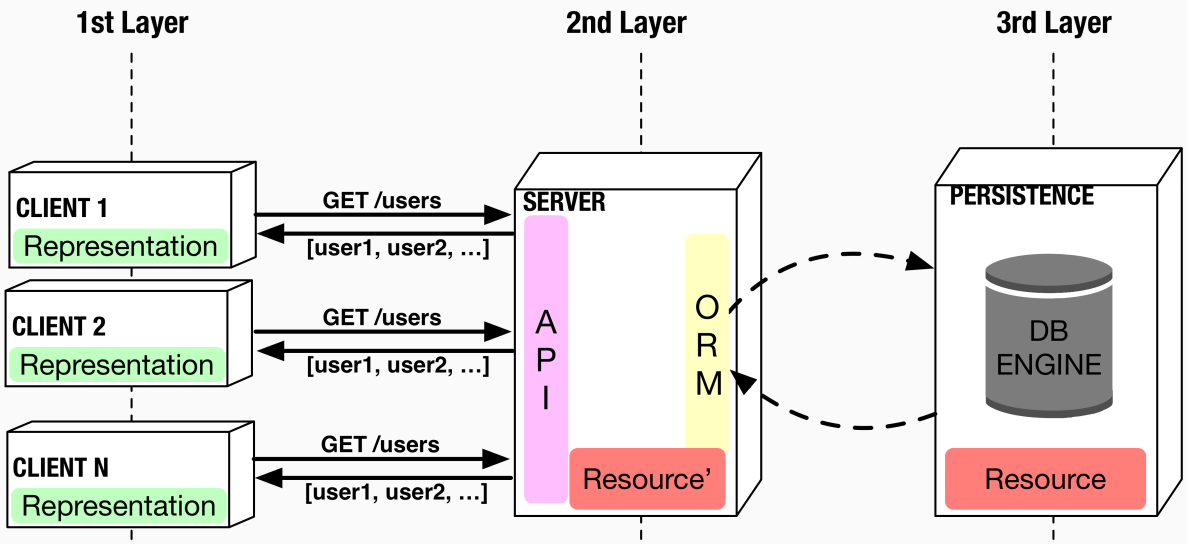
Trade-offs

Uniform Interface : Advantages



- Simplified Architecture
- Improved visibility of interactions
- Decoupled components
- Improved interoperability: Reduces the cost of integration.
- Enables autonomous service discovery

Uniform Interface : Disadvantages



- Degraded efficiency

Abstract

1. Logros
2. Introducción
3. Uniform Interface
4. Trade-offs

Preguntas