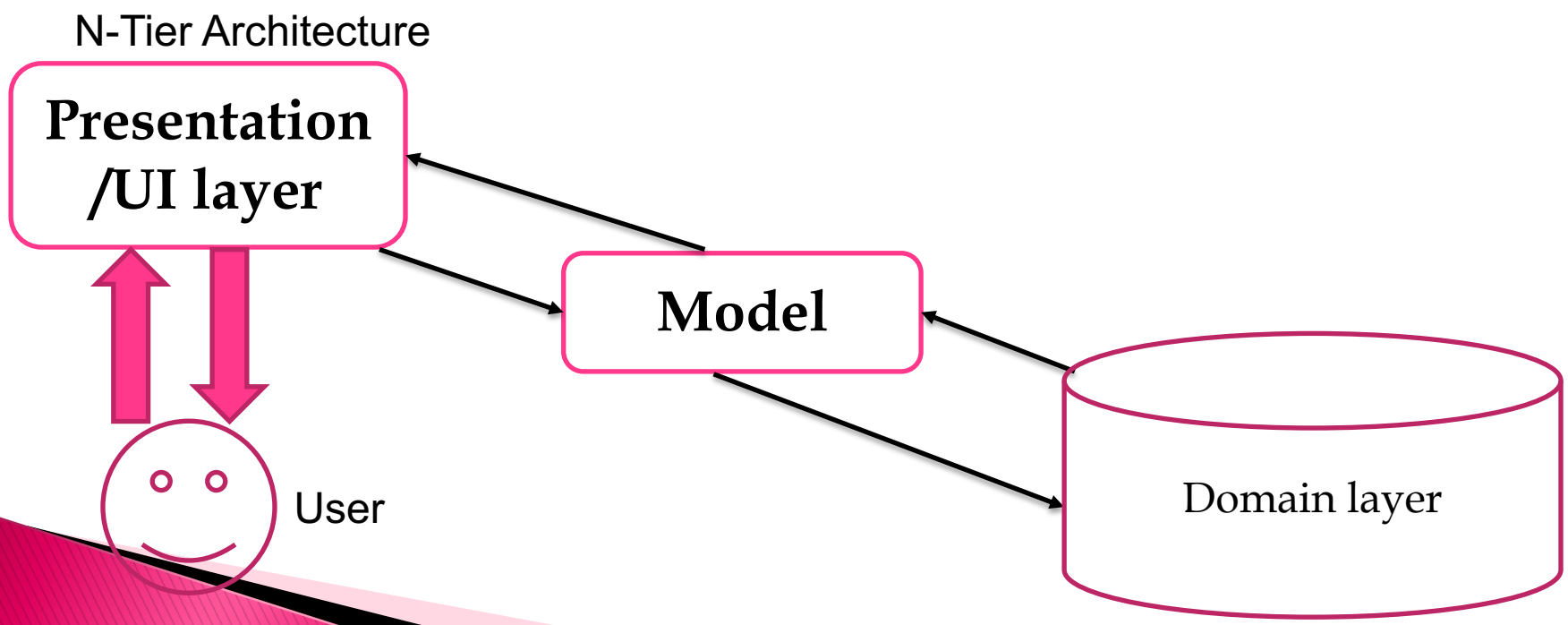
New 8-10-2022

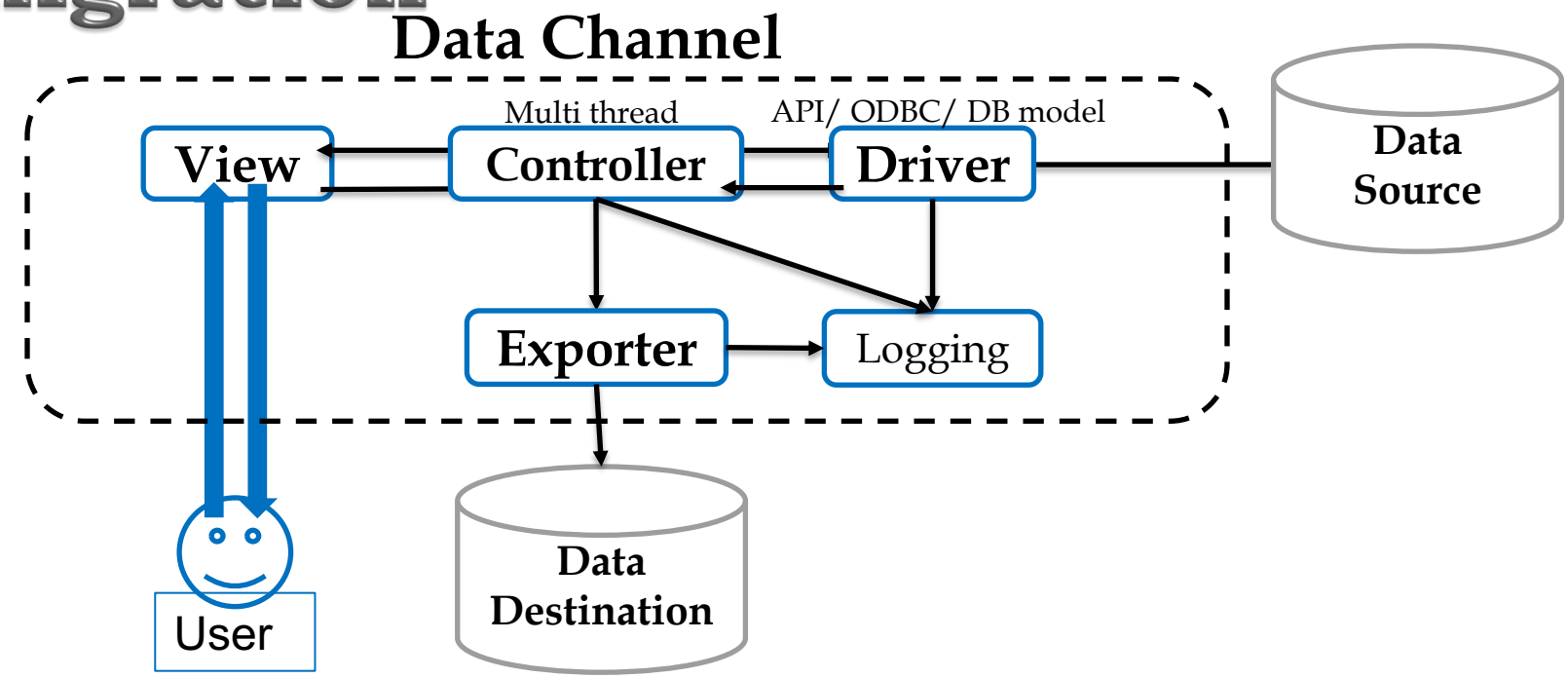
2. Customized architecture

Logic/Domain layer: is a collection of dynamic stored procedures that can rebuild their behavior at run time based on parameters passed to them. Each business entity has one stored procedure that gathers all its operations.



modularized architecture for Data migration

5-Nov-2022



Enterprise components and architectures (Cont.)

- **The SOA infrastructure is composed of the following elements as shown in figure 12:**

1. **Service:** This is essentially a contract: it defines all interfaces and the pre- and postconditions.
2. **Provider:** This is the software entity that implements the service; it accepts and executes requests from consumers.
3. **Consumer** (or requester or client): This is the software entity which calls a service provider to request a service.
4. **Registry** (or locator): This is a software entity, which allows the lookup of services, service providers and their location – in other words it allows the service to be found.

XML vs JSON

- Both are ways for passing values in form of (key, Value) pair allowing recursive data structures.
- JSON: JavaScript Object Notation.
- JSON is text, written with JavaScript object notation.

```
<employees>
  <employee>
    <firstName>John</firstName>
 <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName>
 <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName
> <lastName>Jones</lastName>
  </employee>
</employees>
```

```
{
    "Name": "crunchify.com",
    "Author": "App Shah",
    "Company List": [
        "Compnay: eBay",
        "Compnay: Paypal",
        "Compnay: Google"
    ]
}
```

Example1: Calling web API

```
C:\>>pip install requests
```

Making a Get request

```
# importing the requests library
import requests

# api-endpoint
URL = "http://maps.googleapis.com/maps/api/geocode/json"

# location given here
location = “Suez university"

# defining a params dict for the parameters to be sent to the API
PARAMS = {'address':location}

# sending get request and saving the response as response object
r = requests.get(url = URL, params = PARAMS)
```

https://www.geeksforgeeks.org/get-post-requests-using-python/

Creating a web API in Python

- Don't we need a web server to let the API settle on it?
- How can we direct our calls to the Web server and receive responses?

pip install flask

- Flask: is a lightweight web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications.

https://flask.palletsprojects.com/en/1.1.x/quickstart/#a-minimal-application

Ex1. Quickstart to flask

```
1. from flask import Flask
2. app = Flask(__name__)
3. @app.route('/')
4. def hello_world():
5.     return 'Hello, World!’
```

6. WebHost at : abc.com

Ex1.2. Quickstart to flask

```
1. from flask import Flask
2. app = Flask(__name__)
3. @app.route(‘/helloAgain')
4. def hello_again():
5.     return 'Hello, World!’
```

6. WebHost at : abc.com/helloAgain

QuickStart to flask

1. First we imported the Flask class.
2. Next we create an instance of flask application. The first argument is the name of the application's module or package.
3. We then use the route() decorator to tell Flask what URL should trigger our function.
4. The function is given a name which is also used to generate URLs for that particular function and returns the message we want to display in the user's browser.

QuickStart to flask

5. To run the application

```
C:\path\to\app>set FLASK_APP=hello.py
```

6. Now head over to http://127.0.0.1:5000/, and you should see your hello world greeting.

Ex2. API reading from DB

```
from flask import Flask, request
from flask_restful import Resource, Api
from sqlalchemy import create_engine
from json import dumps
from flask.ext.jsonpify import jsonify

db_connect = create_engine('sqlite:///chinook.db')
app = Flask(__name__)
api = Api(app)

class Employees(Resource):
    def get(self):
        conn = db_connect.connect() # connect to database
        query = conn.execute("select * from employees") # This line performs query and returns json result
        return {'employees': [i[0] for i in query.cursor.fetchall()]} # Fetches first column that is Employee ID
```

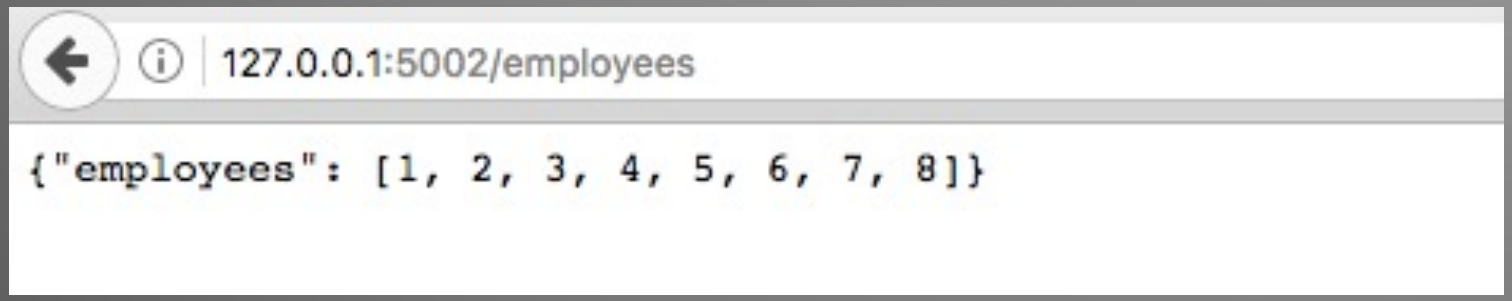
Ex2. API reading from DB

```
class Tracks(Resource):
   def get(self):
      conn = db_connect.connect()
      query = conn.execute("select trackid, name, composer, unitprice from tracks;")
      result = {'data': [dict(zip(tuple (query.keys()) ,i)) for i in query.cursor]}
      return jsonify(result)

class Employees_Name(Resource):
   def get(self, employee_id):
      conn = db_connect.connect()
      query = conn.execute("select * from employees where EmployeeId =%d "  %int(employee_id))
      result = {'data': [dict(zip(tuple (query.keys()) ,i)) for i in query.cursor]}
      return jsonify(result)

api.add_resource(Employees, '/employees') # Route_1
api.add_resource(Tracks, '/tracks') # Route_2
api.add_resource(Employees_Name, '/employees/<employee_id>') # Route_3

if __name__ == '__main__':
    app.run(port='5002')
```

Ex2. output

http://127.0.0.1:5002/employees

shows ids of all the employees in database



Ex2. output

Dr. Hussien M. Sharaf

http://127.0.0.1:5002/tracks *shows tracks details*

127.0.0.1:5002/tracks


```
{
  "data": [
    {
      "Composer": "Angus Young, Malcolm Young, Brian Johnson",
      "Name": "For Those About To Rock (We Salute You)",
      "TrackId": 1,
      "UnitPrice": 0.99
    },
    {
      "Composer": null,
      "Name": "Balls to the Wall",
      "TrackId": 2,
      "UnitPrice": 0.99
    },
    {
      "Composer": "F. Baltes, S. Kaufman, U. Dirkscneider & W. Hoffman",
      "Name": "Fast As a Shark",
      "TrackId": 3,
      "UnitPrice": 0.99
    },
```

Ex2. output

Dr. Hussien M. Sharaf

http://127.0.0.1:5002/employees/8 *shows details of employee whose employeeid is 8*

127.0.0.1:5002/employees/8


```
{
  "data": [
    {
      "Address": "923 7 ST NW",
      "BirthDate": "1968-01-09 00:00:00",
      "City": "Lethbridge",
      "Country": "Canada",
      "Email": "laura@chinookcorp.com",
      "EmployeeId": 8,
      "Fax": "+1 (403) 467-8772",
      "FirstName": "Laura",
      "HireDate": "2004-03-04 00:00:00",
      "LastName": "Callahan",
      "Phone": "+1 (403) 467-3351",
      "PostalCode": "T1H 1Y8",
      "ReportsTo": 6,
      "State": "AB",
      "Title": "IT Staff"
    }
  ]
}
```

EX3: Creating a Short URL API

Endpoints and actions of your URL shortener:

Endpoint	HTTP Verb	Request Body	Action
`/`	`GET`		Returns a `Hello, World!` string
`/url`	`POST`	Your target URL	Shows the created `url_key` with additional info, including a `secret_key`
`/{url_key}`	`GET`		Forwards to your target URL
`/admin/{secret_key}`	`GET`		Shows administrative info about your shortened URL
`/admin/{secret_key}`	`DELETE`	Your secret key	Deletes your shortened URL

https://realpython.com/build-a-python-url-shortener-with-fastapi/

EX3: Creating a Short URL API

Endpoints and actions of your URL shortener:

```
# shortener_app/main.py
import secrets
import validators
from fastapi import Depends, FastAPI, HTTPException

from sqlalchemy.orm import Session
from . import models, schemas
from .database import SessionLocal, engine

app = FastAPI()
models.Base.metadata.create_all(bind=engine)

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

https://realpython.com/build-a-python-url-shortener-with-fastapi/

EX3: Router for URL shortener:

Router for URL shortener:

```
@app.post("/url")
def create_url(url: schemas.URLBase):
    if not validators.url(url.target_url):
        raise_bad_request(message="Your provided URL is not valid")
    return f"TODO: Create database entry for: {url.target_url}“

def raise_not_found(request):
    message = f"URL '{request.url}' doesn't exist"
    raise HTTPException(status_code=404, detail=message)
```

https://realpython.com/build-a-python-url-shortener-with-fastapi/

EX3: Router for URL shortener:

```
@app.post("/url", response_model=schemas.URLInfo)
def create_url(url: schemas.URLBase, db: Session = Depends(get_db)):
    if not validators.url(url.target_url):
        raise_bad_request(message="Your provided URL is not valid")

    chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    key = "".join(secrets.choice(chars) for _ in range(5))
    secret_key = "".join(secrets.choice(chars) for _ in range(8))
    db_url = models.URL(
        target_url=url.target_url, key=key, secret_key=secret_key
    )
    db.add(db_url)
    db.commit()
    db.refresh(db_url)
    db_url.url = key
    db_url.admin_url = secret_key

    return db_url
```

https://realpython.com/build-a-python-url-shortener-with-fastapi/

EX3: Router for URL shortener:

```
@app.get("/{url_key}")
def forward_to_target_url(
        url_key: str,
        request: Request,
        db: Session = Depends(get_db)
    ):
    db_url = (
        db.query(models.URL)
        .filter(models.URL.key == url_key, models.URL.is_active)
        .first()
    )
    if db_url:
        return RedirectResponse(db_url.target_url)
    else:
        raise_not_found(request)
```

https://realpython.com/build-a-python-url-shortener-with-fastapi/

EX3: Router for URL shortener:

```
@app.post("/url", response_model=schemas.URLInfo)
def create_url(url: schemas.URLBase, db: Session = Depends(get_db)):
    if not validators.url(url.target_url):
        raise_bad_request(message="Your provided URL is not valid")

    chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    key = "".join(secrets.choice(chars) for _ in range(5))
    secret_key = "".join(secrets.choice(chars) for _ in range(8))
    db_url = models.URL(
        target_url=url.target_url, key=key, secret_key=secret_key
    )
    db.add(db_url)
    db.commit()
    db.refresh(db_url)
    db_url.url = key
    db_url.admin_url = secret_key

    return db_url
```

https://realpython.com/build-a-python-url-shortener-with-fastapi/

EX3: models.py:

```
# shortener_app/models.py

from sqlalchemy import Boolean, Column, Integer, String

from .database import Base

class URL(Base):
    __tablename__ = "urls"

    id = Column(Integer, primary_key=True)
    key = Column(String, unique=True, index=True)
    secret_key = Column(String, unique=True, index=True)
    target_url = Column(String, index=True)
    is_active = Column(Boolean, default=True)
    clicks = Column(Integer, default=0)
```

https://realpython.com/build-a-python-url-shortener-with-fastapi/

EX3: database.py:

```
# shortener_app/database.py

from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

from .config import get_settings

engine = create_engine(
    get_settings().db_url, connect_args={"check_same_thread": False}
)
SessionLocal = sessionmaker(
    autocommit=False, autoflush=False, bind=engine
)
Base = declarative_base()
```

https://realpython.com/build-a-python-url-shortener-with-fastapi/

```
def main():
    from shortener_app.database import SessionLocal
    db = SessionLocal()
    from shortener_app.models import URL
    db.query(URL).all()

if __name__ == '__main__':
    main()
```

Calling web API

Calling the above code:

```
# extracting data in json format
data = r.json()


# extracting latitude, longitude and formatted address
# of the first matching location
latitude = data['results'][0]['geometry']['location']['lat']
longitude = data['results'][0]['geometry']['location']['lng']
formatted_address = data['results'][0]['formatted_address']

# printing the output
print("Latitude:%s\nLongitude:%s\nFormatted Address:%s"
    %(latitude, longitude,formatted_address))
```

https://www.geeksforgeeks.org/get-post-requests-using-python/

Calling web API

Output:

```
Latitude:28.7499867
Longitude:77.1183137
Formatted Address:Delhi Technological University, Shahbad Daulatpur Village, Rohini, Delhi, 110042, India
```

https://www.geeksforgeeks.org/get-post-requests-using-python/

Example2: Calling web API

Speed test:

```
pip install speedtest-cli
```

```
import speedtest
def test():
    s = speedtest.Speedtest()
    s.get_servers()
    s.get_best_server()
    s.download()
    s.upload()
    res = s.results.dict()
    return res["download"], res["upload"], res["ping"]
```

https://stackoverflow.com/questions/48289636/speedtest-python-script

Calling web API continue

```
def main():
    # simply print in needed format if you want to use pipe-style:
    python script.py > file
    for i in range(3):
        d, u, p = test()
        print('Test #{}\n'.format(i+1))
        print('Download: {:.2f} Kb/s\n'.format(d / 1024))
        print('Upload: {:.2f} Kb/s\n'.format(u / 1024))
        print('Ping: {}\n'.format(p))


if __name__ == '__main__':
    main()
```

https://stackoverflow.com/questions/48289636/speedtest-python-script