Devon4Py

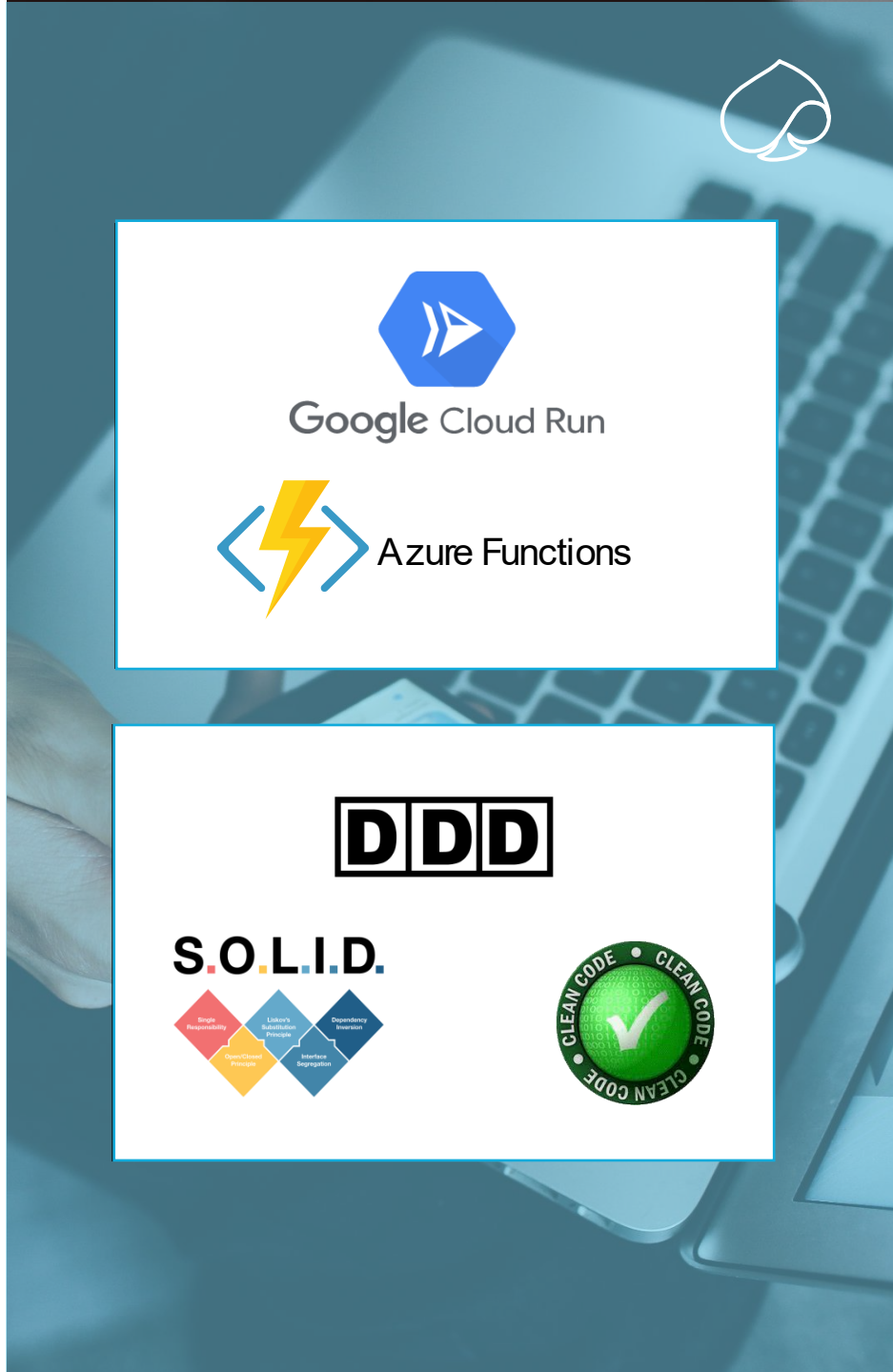
ADCenter

Services

Capgemini

Python Incubator

- Scalable & Easy to Maintain Template
- Python development standardization
- Production Ready for API applications
- Industrialized & Standards-Based



Industrialization

Service Layer

Set of *classes* or *functions*, called ***Services***
that together form an **API**
for a single package or application

Service

provides an indivisible ***piece of functionality***
for an actor
using the system

Add an *item* to a *basket*
User Pay an *order*

Industrialization

Service Layer

Set of *classes* or *functions*, called ***Services***
that together form an **API**
for a single package or application

Aims to **provide** a **single place to look** for **application logic**

Ignores the **actor interface** and the **delivery mechanism**

Helps to **bridge** the gap **between** **business** and **software** development

Consumer, Service & Persistence

Business

Controller

Consumers
e.g.: Events, CLI, HTTP Request

Service

Business Rules

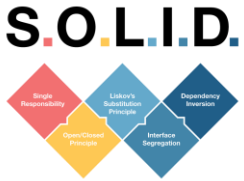
Application Logic

Domain

Repository

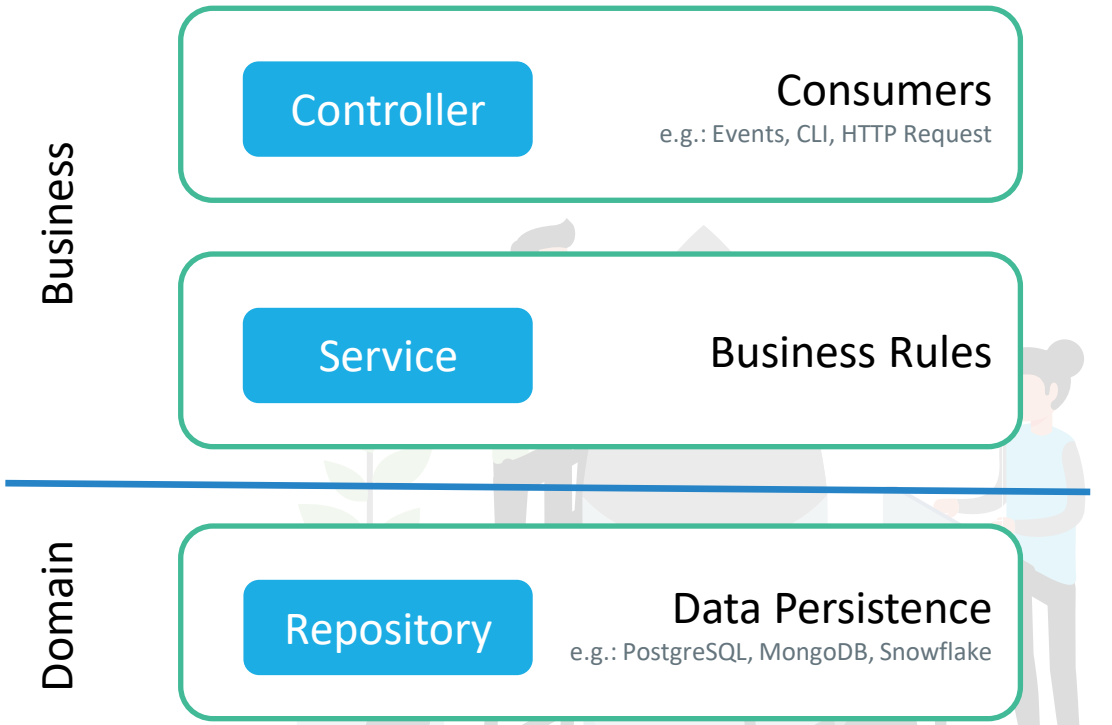
Data Persistence
e.g.: PostgreSQL, MongoDB, Snowflake

Development
Standarization

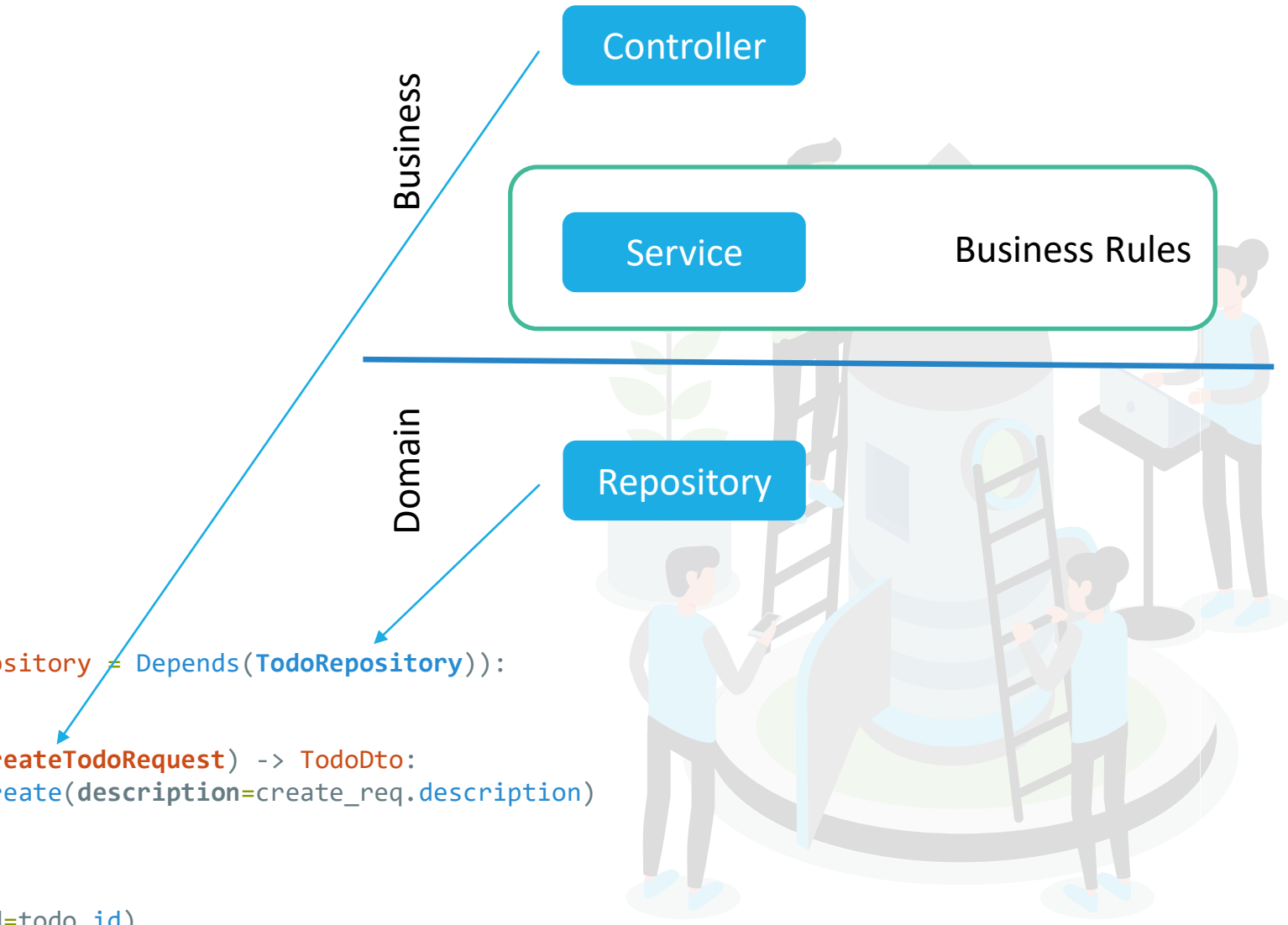


devon4py Scaffolding

- *My devon4py app*
  - **Business**
    - **Controllers**
    - **Services**
    - Models (Requests, Responses & DTOs)
  - **Domain**
    - **Repositories**
    - Models (Entities)
  - **Common Core**



devon4py Scaffolding

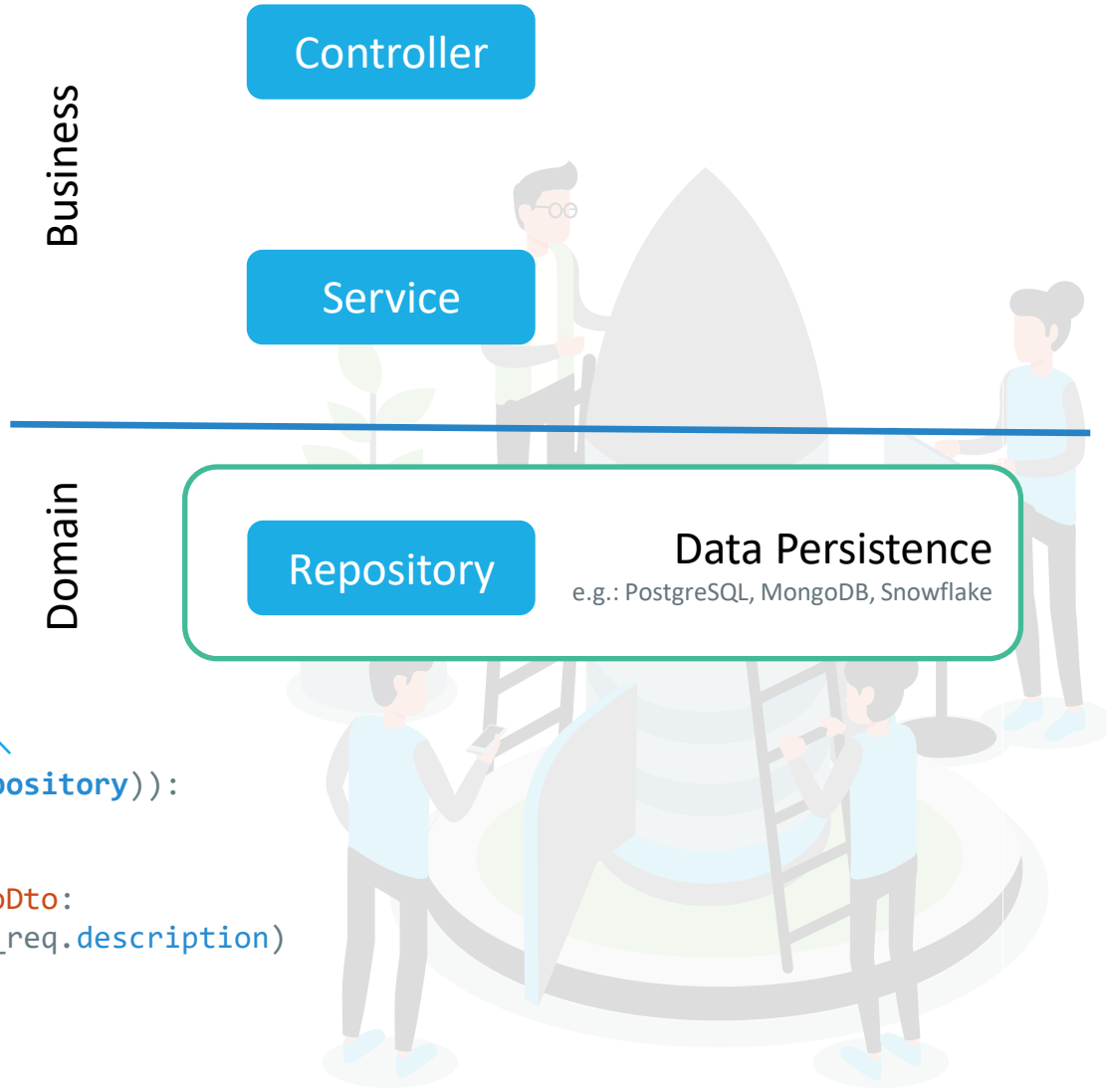



```python
class TodoService:
    def __init__(self, repository: TodoSQLRepository = Depends(TodoRepository)):
        self.todo_repo = repository

    async def create_todo(self, create_req: CreateTodoRequest) -> TodoDto:
        raw_new_todo = await self.todo_repo.create(description=create_req.description)
        return parse_to_dto(raw_new_todo)

    async def todo_done(self, todo: TodoID):
        await self.todo_repo.todo_done(todo_id=todo.id)
```

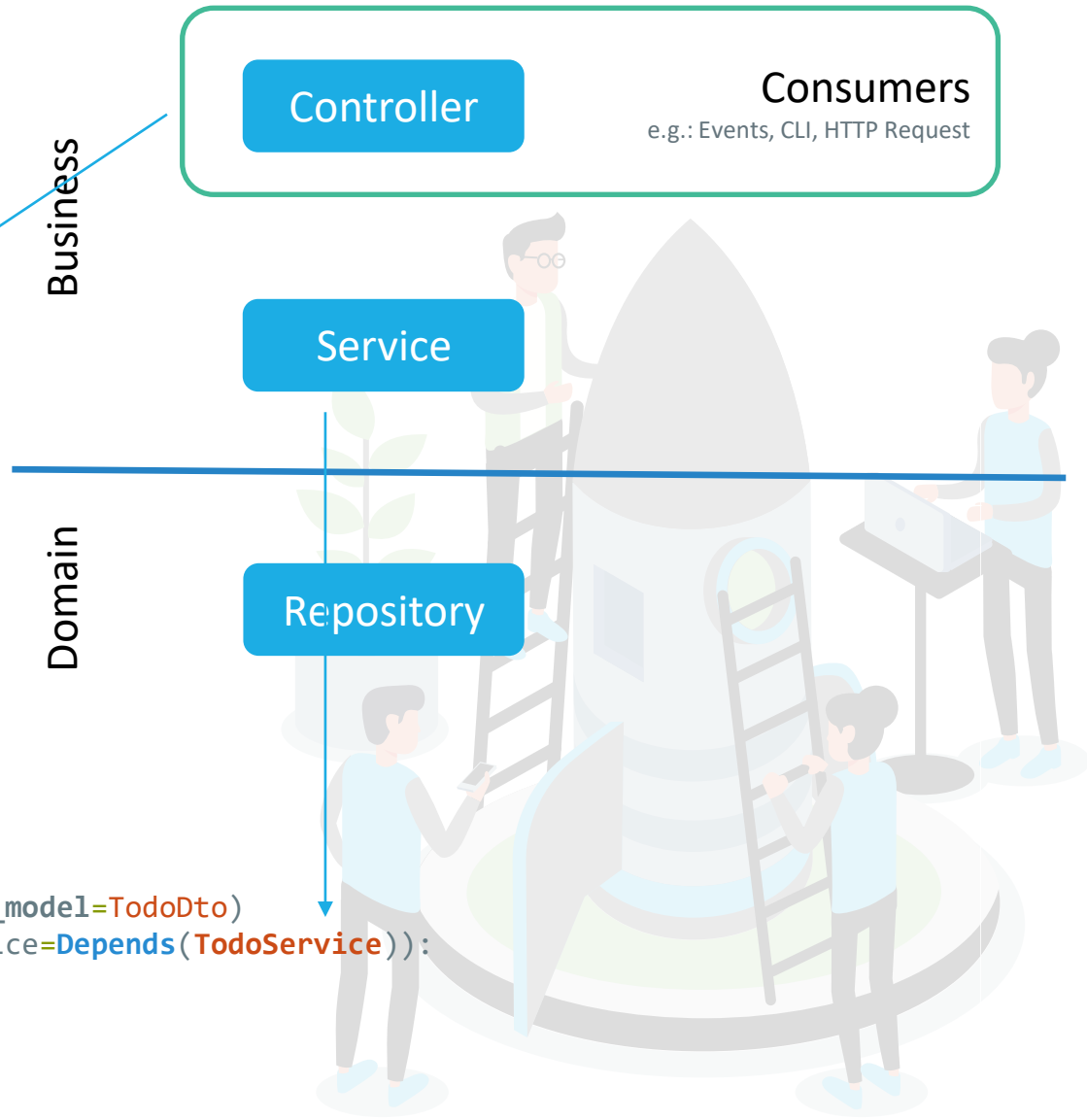
devon4py Scaffolding

```python
class TodoSQLRepository(BaseSQLRepository[Todo]):

    async def create(self, *, description: str) -> Todo:
        new_todo = Todo(description=description)
        await self.add(model=new_todo)
        return new_todo

    async def get_pending_todos(self) -> List[Todo]:
        todos = await self.session.exec(
           select(Todo).where(Todo.done == False)
        )
        return todos.all()
```

```python
class TodoService:
    def __init__(self, repository: TodoSQLRepository = Depends(TodoRepository)):
        self.todo_repo = repository

    async def create_todo(self, create_req: CreateTodoRequest) -> TodoDto:
        raw_new_todo = await self.todo_repo.create(description=create_req.description)
        return parse_to_dto(raw_new_todo)

    async def todo_done(self, todo: TodoID):
        await self.todo_repo.todo_done(todo_id=todo.id)
```

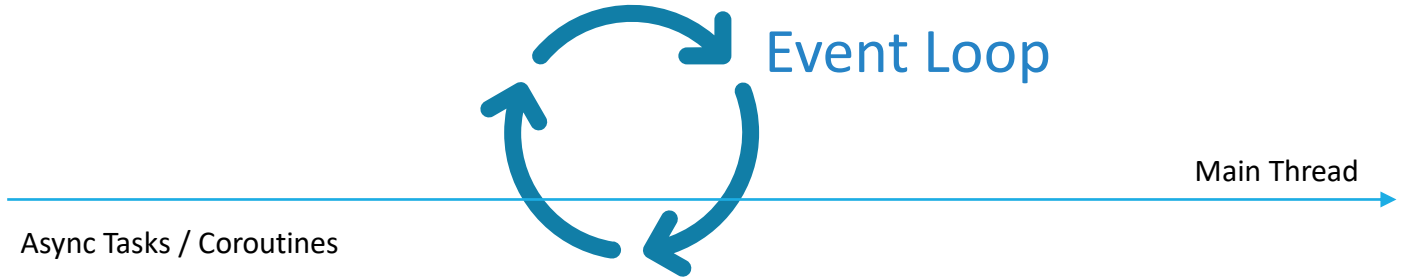


devon4py Scaffolding




```python
@router.post("/create", description="Creates a new TODO", response_model=TodoDto)
async def create_todo(create_request: CreateTodoRequest, todo_service=Depends(TodoService)):
    todo = await todo_service.create_todo(create_request)
    return todo
```

Concurrency and Parallelism




```python
async def create_todo(create_request: CreateTodoRequest, todo_service=Depends(TodoService)):
    await todo_service.create_todo(create_request)
```

Main Thread can handle multiple requests and schedule them on Event Loop

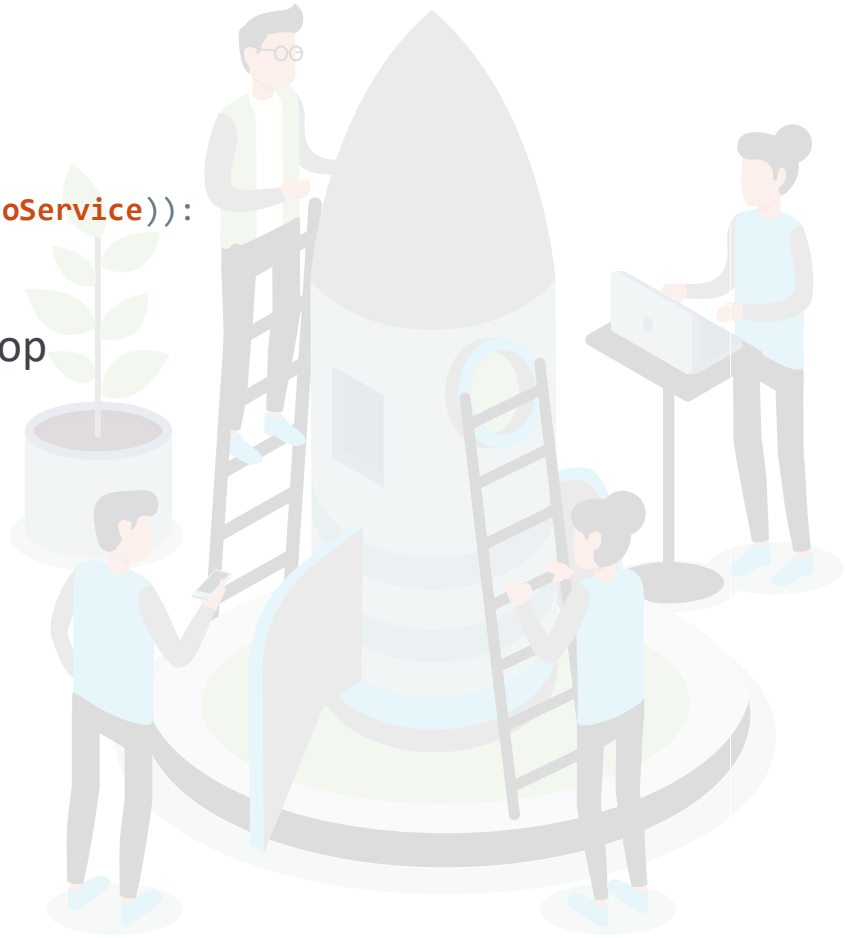
For I/O Intensive Tasks

Non Async Tasks will be handled in a Thread Pool by a separate Thread

For Compute Intensive Tasks

Multiple processes “workers” can also be launched in parallel

```python
uvicorn.run("app.api:api", workers=4)
```



Dependencies & Modules

Pydantic

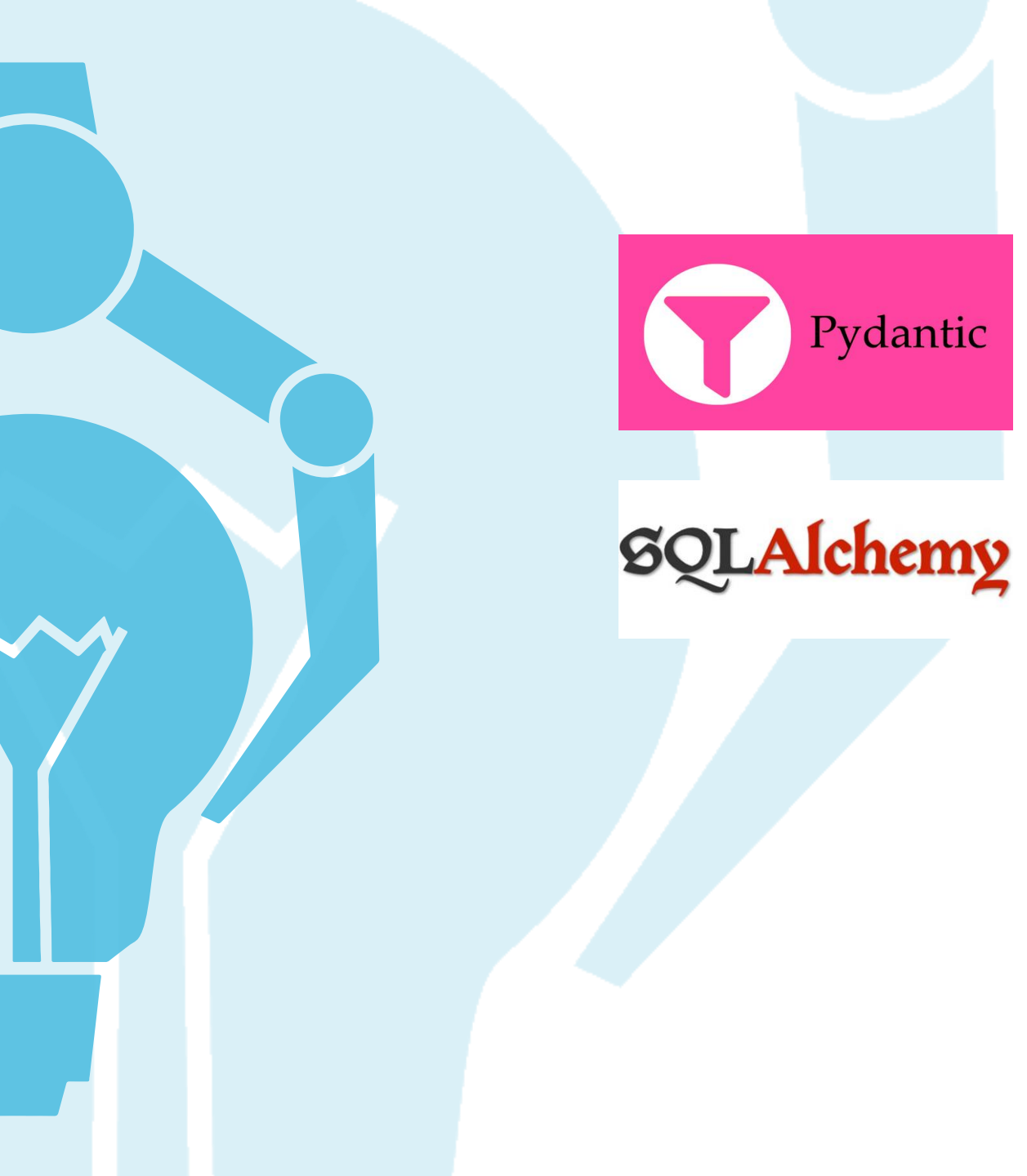
Data Validation & Settings Management

Enforces ***type hints*** at runtime and provides user friendly errors when data is invalid

Define how data should be in pure, canonical python; validate it with ***pydantic***

```python
class User(BaseModel):
    id: int
    signup_ts: Optional[datetime] = None
    friends: List[int] = []


external_data = {
    'id': '123',
    'signup_ts': '2019-06-01 12:22',
    'friends': [1, 2, '3'],
}

user = User(**external_data)
user = User(id=12, signup_ts=datetime.now(), friends=[13, 14])
```

Dependencies & Modules



Pydantic

SQLAlchemy

Data Validation & Settings Management

Enforces ***type hints*** at runtime and provides user friendly errors when data is invalid

Define how data should be in pure, canonical python; validate it with ***pydantic***

SQL Toolkit and Object Relational Mapper

Full suite of enterprise-level persistence patterns,

designed for efficient and high-performing database access

Dependencies & Modules

Pydantic

Data Validation & Settings Management

Enforces ***type hints*** at runtime and provides user friendly errors when data is invalid

Define how data should be in pure, canonical python; validate it with ***pydantic***

SQLAlchemy

SQL Toolkit and Object Relational Mapper

Full suite of enterprise-level persistence patterns,

designed for efficient and high-performing database access

SQLModel

Dependencies & Modules

```python
from typing import Optional

from sqlmodel import Field, Session, SQLModel, create_engine, select


class Hero(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)
    name: str
    secret_name: str
    age: Optional[int] = None


engine = create_engine("sqlite:///database.db")

with Session(engine) as session:
    statement = select(Hero).where(Hero.name == "Spider-Boy")
    hero = session.exec(statement).first()
    print(hero.)
```

```
name
age
Config
construct
copy
dict
from_orm
id
json
metadata
```

Management

y errors when data is invalid

on; validate it with ***pydantic***

onal Mapper

e-level persistence patterns,

performing database access

Dependencies & Modules

Data Validation & Settings Management

Enforces ***type hints*** at runtime and provides user friendly errors when data is invalid

Define how data should be in pure, canonical python; validate it with ***pydantic***

SQL Toolkit and Object Relational Mapper

Full suite of enterprise-level persistence patterns,

designed for efficient and high-performing database access

ASGI Framework / toolkit

Lightweight & low-complexity HTTP web framework

Optimized for building **async** web services

Dependencies & Modules

Rnk	Framework	Best performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	uvicorn	71,609 100.0% (10.7%)	0	Plt	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
2	apidaora-core	70,241 98.1% (10.5%)	0	Mcr	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
3	blacksheep	68,020 95.0% (10.2%)	0	Plt	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
4	sanic	64,442 90.0% (9.7%)	0	Mcr	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
5	apidaora	62,849 87.8% (9.4%)	0	Mcr	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
6	starlette	61,334 85.7% (9.2%)	0	Mcr	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
7	fastapi	52,095 72.7% (7.8%)	0	Mcr	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
8	fastapi-orjson	51,867 72.4% (7.8%)	0	Mcr	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
9	responder	43,542 60.8% (6.5%)	0	Plt	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
10	aiohttp-pg-raw	36,333 50.7% (5.4%)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
11	bottle-raw	35,047 48.9% (5.3%)	0	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea
12	api_hour	26,683 37.3% (4.0%)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
13	quart-uvicorn	26,446 36.9% (4.0%)	0	Mcr	Py	Non	uvi	Lin	Pg	Lin	Raw	Rea
14	flask-raw	23,573 32.9% (3.5%)	0	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea
15	aiohttp	22,561 31.5% (3.4%)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Ful	Rea
16	pyramid-py2	21,493 30.0% (3.2%)	0	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea
17	tornado-py3-uvloop	20,538 28.7% (3.1%)	0	Plt	Py	Non	Tor	Lin	Pg	Lin	Raw	Rea
18	api_hour-mysql	19,846 27.7% (3.0%)	0	Mcr	Py	asy	Gun	Lin	My	Lin	Raw	Rea
19	pyramid	19,608 27.4% (2.9%)	0	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea
20	flask-pypy2-raw	18,121 25.3% (2.7%)	0	Mcr	Py	Tor	Non	Lin	My	Lin	Raw	Rea

Dependencies & Modules

Data Validation & Settings Management

Enforces ***type hints*** at runtime and provides user friendly errors when data is invalid

Define how data should be in pure, canonical python; validate it with ***pydantic***

SQL Toolkit and Object Relational Mapper

Full suite of enterprise-level persistence patterns,

designed for efficient and high-performing database access

ASGI Framework / toolkit

Lightweight & low-complexity HTTP web framework

Optimized for building **async** web services

ASGI Web Server

Cython based dependencies

Using ***uvloop***, a fast replacement of the built-in asyncio event loop

High-level Python objects wrap low-level **libuv** structs and functions

Microsoft

Uber

NETFLIX

aws

FastAPI Short, fast to code & Intuitive

Dependencies & Modules

Data Validation & Settings Management

Pydantic

Enforces ***type hints*** at runtime and provides user friendly errors when data is invalid

Define how data should be in pure, canonical python; validate it with ***pydantic***

SQL Toolkit and Object Relational Mapper

SQLAlchemy

SQLModel

Full suite of enterprise-level persistence patterns,

designed for efficient and high-performing database access

ASGI Framework / toolkit

Starlette

Lightweight & low-complexity HTTP web framework

Optimized for building **async** web services

ASGI Web Server

Cython based dependencies

Using ***uvloop***, a fast replacement of the built-in asyncio event loop

High-level Python objects wrap low-level **libuv** structs and functions

Devon4py

- **Production-ready** Template
- Industrialization
- Standarization

- **SOLID**
- Domain Driven Development
- Clean Code

- **Google Services Integration**
- Ready to deploy as **Serverless** service on:

Google Cloud Run

Azure Functions

Starlette

Micro-framework / Web Toolkit

Best performance for async web services

Uvicorn

ASGI web server implementation

Best performance for async web server

Uvloop

High-performance asyncio

2x faster than Nodejs, close to Go programs

Cython

Compiled Python

C-Extensions for Python

FastAPI

Fast to code increase speed development 200% - 300%

WEB Framework with **Dependency Injection, Easy & Short**

SQL Model

Reduce code duplication

Best developer experience

Pydantic

Data Validation & Serialization

Type Annotations & Settings Management

Cython

Compiled Python

C-Extensions for Python

SQL Alchemy

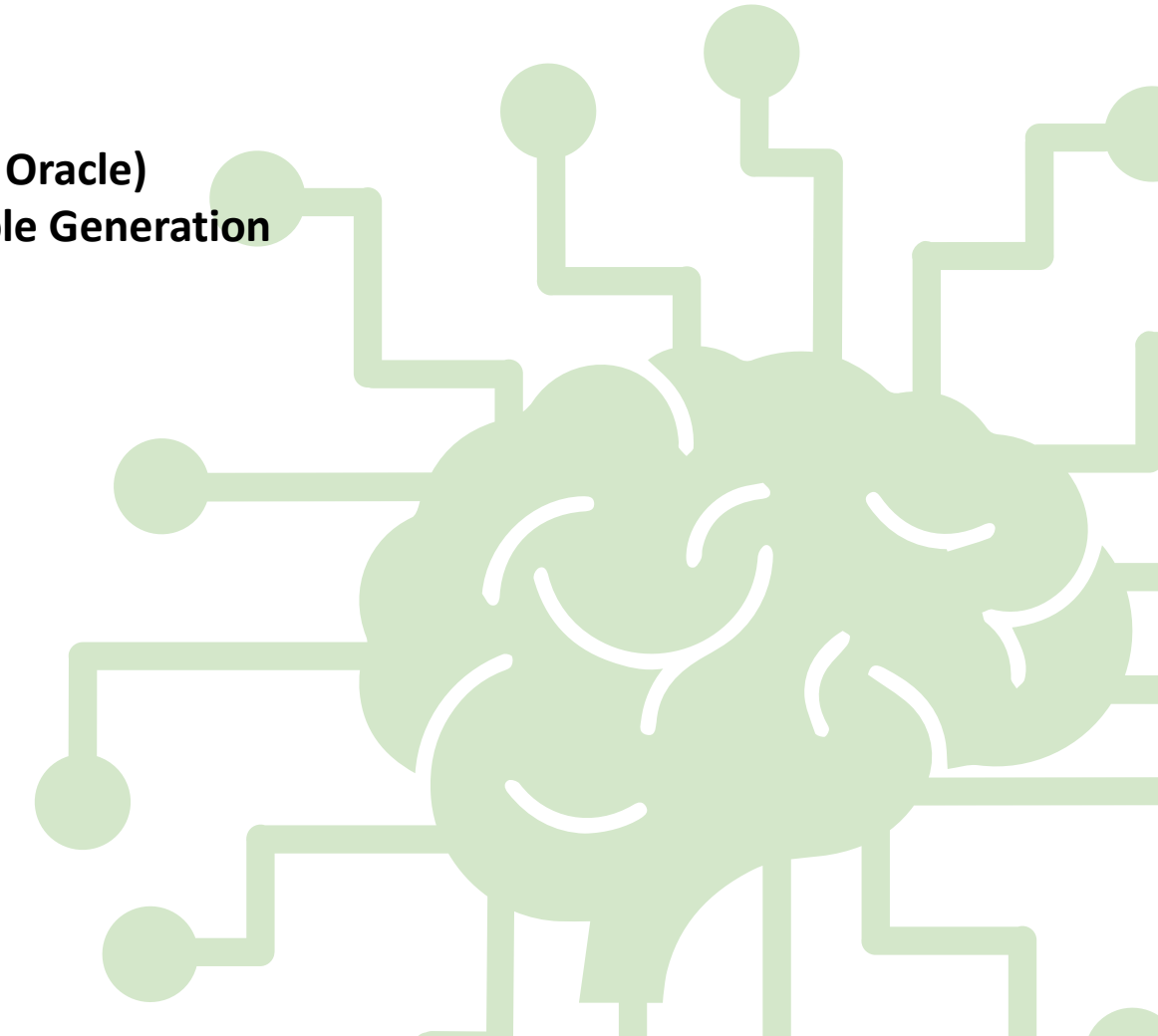
SQL Toolkit

Object Relational Mapper

Roadmap

Included in the solution

- **Swagger** / OpenAPI integration
- **Configuration** file **per environment**
- **Log** Configuration + **Files Sink & GCloud Logging** Adaptor
- **CORS Middleware** Configuration
- **Database** Configuration + **SQL Adaptors (Snowflake, PG & Oracle)**
- **ORM** Integration + Base Repository + Automatic **Entity Table Generation**
- **Controller – Service – Repository** Integration
- JWT **Authorization** / **Keycloak** Integration
- Global **Exception Management**
- **Server-sent Events**
- **Lambda Azure** Functions Integration
- **Google Firebase Client** Integration
- **Dynamic Adaptors**
- **Firestore DB** Client + Base Repository
- **Firebase Storage** Client Integration
- **Firebase Authentication** Identity Provider
- **Firebase Push Notifications** Service (GCloud Messaging)
- **Documentation** & **Onboarding** (ToDo & JumpTheQueue**)**



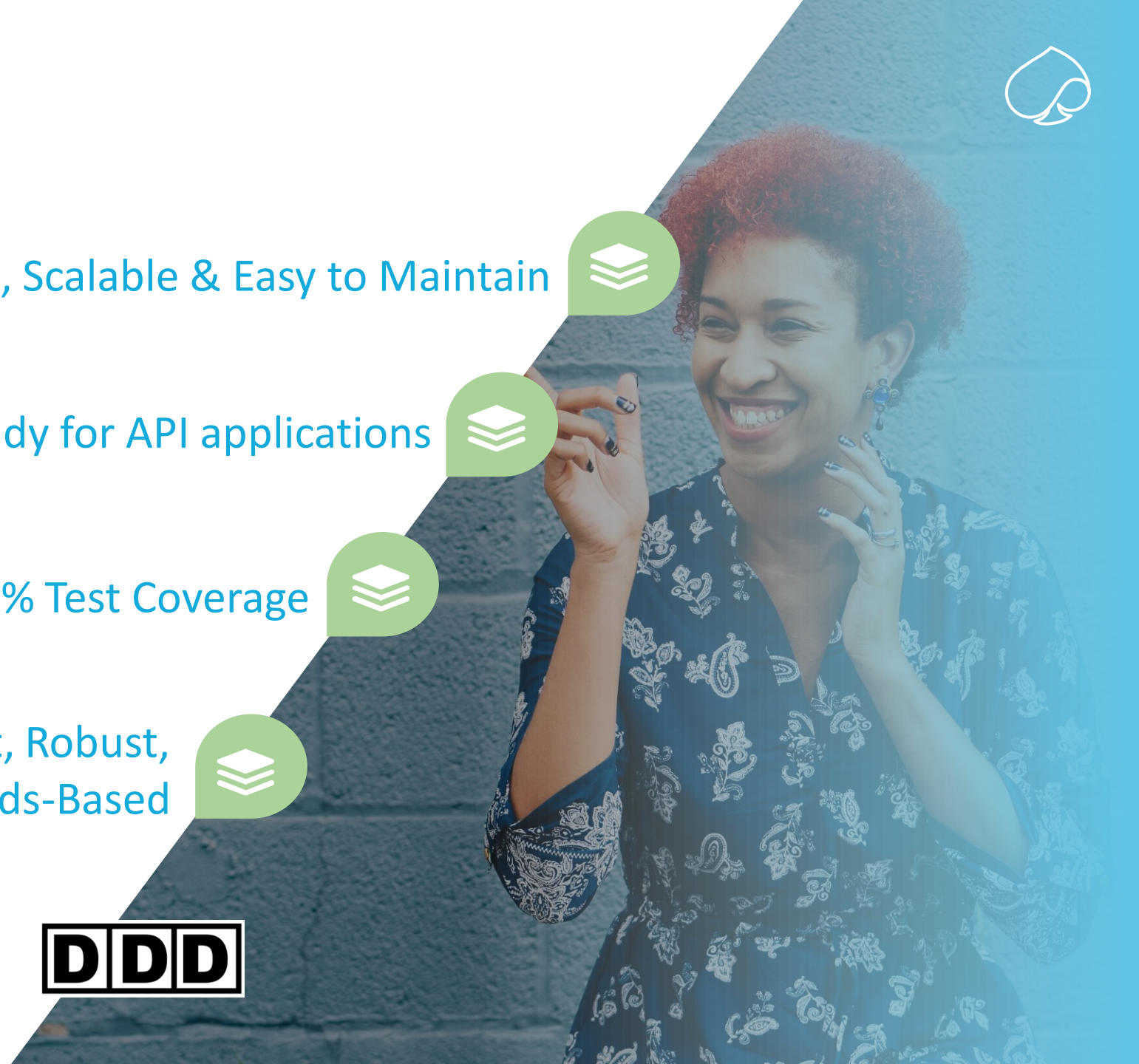
devon4Py

Very High Performance, Scalable & Easy to Maintain

Production Ready for API applications

Type Annotated & Possibility of 100% Test Coverage

Fast to code, Intuitive, Easy, Short, Robust,
Industrialized & Standards-Based



Development
Standarization

Capgemini

People matter, results count.

About Capgemini

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of 200,000 team members in over 40 countries. The Group reported 2016 global revenues of EUR 12.5 billion.

Learn more about us at

www.capgemini.com

This presentation contains information that may be privileged or confidential and is the property of the Capgemini Group.
Copyright © 2018 Capgemini. All rights reserved.