REST APIs / FastAPI:

Τι είναι το REST API; / What is a REST API?

Το REST (Representational State Transfer) είναι ένα αρχιτεκτονικό στυλ για την ανάπτυξη διαδικτυακών υπηρεσιών. Τα REST APIs χρησιμοποιούν το HTTP πρωτόκολλο και τις μεθόδους του για επικοινωνία.

REST is an architectural style for developing web services. REST APIs use the HTTP protocol and its methods for communication.

Βασικές Αρχές REST / REST Basic Principles

1. **Stateless** - Κάθε αίτημα είναι ανεξάρτητο / Each request is independent
2. **Client-Server** - Διαχωρισμός ευθυνών / Separation of concerns
3. **Uniform Interface** - Συνεπής διεπαφή / Consistent interface
4. **Resource-Based** - Πόροι με μοναδικά URLs / Resources with unique URLs

HTTP Μέθοδοι / HTTP Methods

- GET: Ανάκτηση δεδομένων / Retrieve data
- POST: Δημιουργία νέων δεδομένων / Create new data
- PUT/PATCH: Ενημέρωση δεδομένων / Update data
- DELETE: Διαγραφή δεδομένων / Delete data

Εισαγωγή στο FastAPI / Introduction to FastAPI

Το FastAPI είναι ένα σύγχρονο framework για την ανάπτυξη REST APIs σε Python. Είναι γρήγορο, εύκολο στη χρήση και παρέχει αυτόματη τεκμηρίωση.

FastAPI is a modern framework for building REST APIs in Python. It's fast, easy to use, and provides automatic documentation.

Εγκατάσταση / Installation

```
pip install fastapi uvicorn
```

Βασικά Παραδείγματα / Basic Examples

1. Απλό API / Simple API

```python
from fastapi import FastAPI

# Δημιουργία εφαρμογής / Create application
app = FastAPI()

# Βασικό endpoint / Basic endpoint
@app.get("/")
def read_root():
    return {"message": "Καλώς ήρθατε στο API μας! / Welcome to our API!"}
```

Εκκίνηση server / Start server (terminal)

```
uvicorn main:app --reload
```

2. Path Parameters

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/items/{item_id}")
def read_item(item_id: int):
    return {"item_id": item_id}
```

3. Query Parameters

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/search/")
def search_items(query: str, limit: int = 10):
    return {
        "query": query,
        "limit": limit
    }
```

4. Σύνθετο Παράδειγμα με Pydantic / Complex Example with Pydantic

```python
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import List, Optional

# Ορισμός μοντέλου δεδομένων / Define data model
class Student(BaseModel):
    id: Optional[int] = None
    name: str
    age: int
    courses: List[str]

app = FastAPI()

# Προσομοίωση βάσης δεδομένων / Simulate database
students_db = []

# Δημιουργία φοιτητή / Create student
@app.post("/students/", response_model=Student)
def create_student(student: Student):
    student.id = len(students_db) + 1
    students_db.append(student)
    return student

# Λήψη όλων των φοιτητών / Get all students
@app.get("/students/", response_model=List[Student])
def get_students():
    return students_db

# Λήψη συγκεκριμένου φοιτητή / Get specific student
@app.get("/students/{student_id}", response_model=Student)
```

```python
def get_student(student_id: int):
    if student_id < 1 or student_id > len(students_db):
        raise HTTPException(status_code=404, detail="Student not found")
    return students_db[student_id - 1]
```

5. Παράδειγμα με Error Handling / Example with Error Handling

```python
from fastapi import FastAPI, HTTPException, status
from pydantic import BaseModel

app = FastAPI()

class Item(BaseModel):
    name: str
    price: float

@app.post("/items/")
def create_item(item: Item):
    if item.price < 0:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Η τιμή δεν μπορεί να είναι αρνητική / Price cannot be
negative"
        )
    return item
```

Τεκμηρίωση API / API Documentation

Το FastAPI δημιουργεί αυτόματα τεκμηρίωση στα endpoints: FastAPI automatically generates documentation at these endpoints:

- `/docs` - SwaggerUI (Interactive documentation)
- `/redoc` - ReDoc (Alternative documentation)

Προχωρημένο Παράδειγμα: Todo API / Advanced Example: Todo API

```python
from fastapi import FastAPI, HTTPException, status
from pydantic import BaseModel
from typing import List, Optional

app = FastAPI(title="Todo API")

class TodoItem(BaseModel):
    id: Optional[int] = None
    title: str
    description: str
    completed: bool = False

todos_db = []

@app.post("/todos/", response_model=TodoItem,
status_code=status.HTTP_201_CREATED)
def create_todo(todo: TodoItem):
    """
    Δημιουργία νέου todo / Create new todo
    """
    todo.id = len(todos_db) + 1
```

```
    todos_db.append(todo)
    return todo

@app.get("/todos/", response_model=List[TodoItem])
def list_todos(completed: Optional[bool] = None):
    """
    Λίστα όλων των todos με προαιρετικό φίλτρο / List all todos with
optional filter
    """
    if completed is None:
        return todos_db
    return [todo for todo in todos_db if todo.completed == completed]

@app.get("/todos/{todo_id}", response_model=TodoItem)
def get_todo(todo_id: int):
    """
    Λήψη συγκεκριμένου todo / Get specific todo
    """
    if todo_id < 1 or todo_id > len(todos_db):
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="Todo not found"
        )
    return todos_db[todo_id - 1]

@app.put("/todos/{todo_id}", response_model=TodoItem)
def update_todo(todo_id: int, updated_todo: TodoItem):
    """
    Ενημέρωση todo / Update todo
    """
    if todo_id < 1 or todo_id > len(todos_db):
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="Todo not found"
        )
    todos_db[todo_id - 1] = updated_todo
    updated_todo.id = todo_id
    return updated_todo

@app.delete("/todos/{todo_id}", status_code=status.HTTP_204_NO_CONTENT)
def delete_todo(todo_id: int):
    """
    Διαγραφή todo / Delete todo
    """
    if todo_id < 1 or todo_id > len(todos_db):
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="Todo not found"
        )
    todos_db.pop(todo_id - 1)
    return None
```

In [ ]: