DATABASE

users = {}

houses = {}

floors = {}

rooms = {}

devices = {}

DATA MODELS (CLASSES)

* User:
  + user\_id: str
  + username: str
  + email: Optional[str] = None
* House:
  + house\_id: str
  + address: str
  + gps\_location: str
  + size: float
  + floors: List[str] = []
* Floor:
  + floor\_id: str
  + house\_id: str
  + rooms: List[str] = []
* Room:
  + room\_id: str
  + name: str
  + floor\_id: str
  + devices: List[str] = []
* Devices:
  + device\_id: str
  + type: str
  + room\_id: str

CRUD Operations

USER

* ("/users/")

def create\_user(user: User):

users[user.user\_id] = user

return {"message": "User created", "user": user}

* ("/users/{user\_id}")

def read\_user(user\_id: str):

return users.get(user\_id, {"error": "User not found"})

* ("/users/{user\_id}")

def update\_user(user\_id: str, user: User):

users[user\_id] = user

return {"message": "User updated", "user": user}

* ("/users/{user\_id}")

def delete\_user(user\_id: str):

users.pop(user\_id, None)

return {"message": "User deleted"}

# House Endpoints

* ("/houses/")

def create\_house(house: House):

houses[house.house\_id] = house

return {"message": "House created", "house": house}

* ("/houses/{house\_id}")

def read\_house(house\_id: str):

return houses.get(house\_id, {"error": "House not found"})

* ("/houses/{house\_id}")

def update\_house(house\_id: str, house: House):

houses[house\_id] = house

return {"message": "House updated", "house": house}

* ("/houses/{house\_id}")

def delete\_house(house\_id: str):

houses.pop(house\_id, None)

return {"message": "House deleted"}

# Floor Endpoints

* ("/floors/")

def create\_floor(floor: Floor):

floors[floor.floor\_id] = floor

houses[floor.house\_id].floors.append(floor.floor\_id)

return {"message": "Floor created", "floor": floor}

* ("/floors/{floor\_id}")

def read\_floor(floor\_id: str):

return floors.get(floor\_id, {"error": "Floor not found"})

* ("/floors/{floor\_id}")

def update\_floor(floor\_id: str, floor: Floor):

floors[floor\_id] = floor

return {"message": "Floor updated", "floor": floor}

* ("/floors/{floor\_id}")

def delete\_floor(floor\_id: str):

floors.pop(floor\_id, None)

return {"message": "Floor deleted"}

# Room Endpoints

* ("/rooms/")

def create\_room(room: Room):

rooms[room.room\_id] = room

floors[room.floor\_id].rooms.append(room.room\_id)

return {"message": "Room created", "room": room}

* ("/rooms/{room\_id}")

def read\_room(room\_id: str):

return rooms.get(room\_id, {"error": "Room not found"})

* ("/rooms/{room\_id}")

def update\_room(room\_id: str, room: Room):

rooms[room\_id] = room

return {"message": "Room updated", "room": room}

* ("/rooms/{room\_id}")

def delete\_room(room\_id: str):

rooms.pop(room\_id, None)

return {"message": "Room deleted"}

# Device Endpoints

* ("/devices/")

def create\_device(device: Device):

devices[device.device\_id] = device

rooms[device.room\_id].devices.append(device.device\_id)

return {"message": "Device created", "device": device}

* ("/devices/{device\_id}")

def read\_device(device\_id: str):

return devices.get(device\_id, {"error": "Device not found"})

* ("/devices/{device\_id}")

def update\_device(device\_id: str, device: Device):

devices[device\_id] = device

return {"message": "Device updated", "device": device}

* ("/devices/{device\_id}")

def delete\_device(device\_id: str):

devices.pop(device\_id, None)

return {"message": "Device deleted"}

# Procedure-Based API Operations

* ("/houses/{house\_id}/add\_floor")

def add\_floor\_to\_house(house\_id: str, floor: Floor):

floors[floor.floor\_id] = floor

houses[house\_id].floors.append(floor.floor\_id)

return {"message": "Floor added to house"}

* ("/floors/{floor\_id}/add\_room")

def add\_room\_to\_floor(floor\_id: str, room: Room):

rooms[room.room\_id] = room

floors[floor\_id].rooms.append(room.room\_id)

return {"message": "Room added to floor"}

* ("/houses/{house\_id}/remove\_floor/{floor\_id}")

def remove\_floor\_from\_house(house\_id: str, floor\_id: str):

houses[house\_id].floors.remove(floor\_id)

floors.pop(floor\_id, None)

return {"message": "Floor removed from house"}

* ("/floors/{floor\_id}/remove\_room/{room\_id}")

def remove\_room\_from\_floor(floor\_id: str, room\_id: str):

floors[floor\_id].rooms.remove(room\_id)

rooms.pop(room\_id, None)

return {"message": "Room removed from floor"}