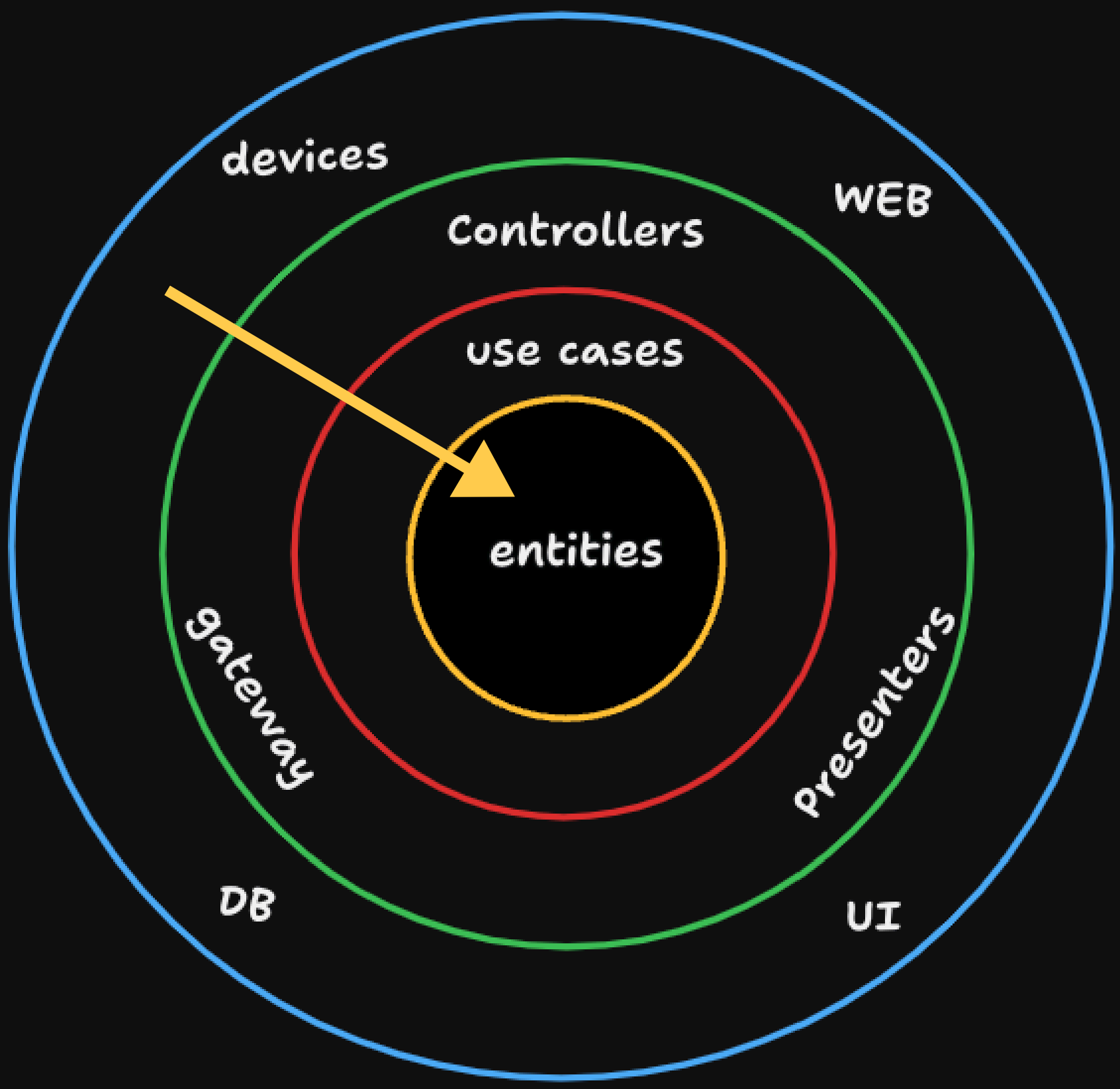
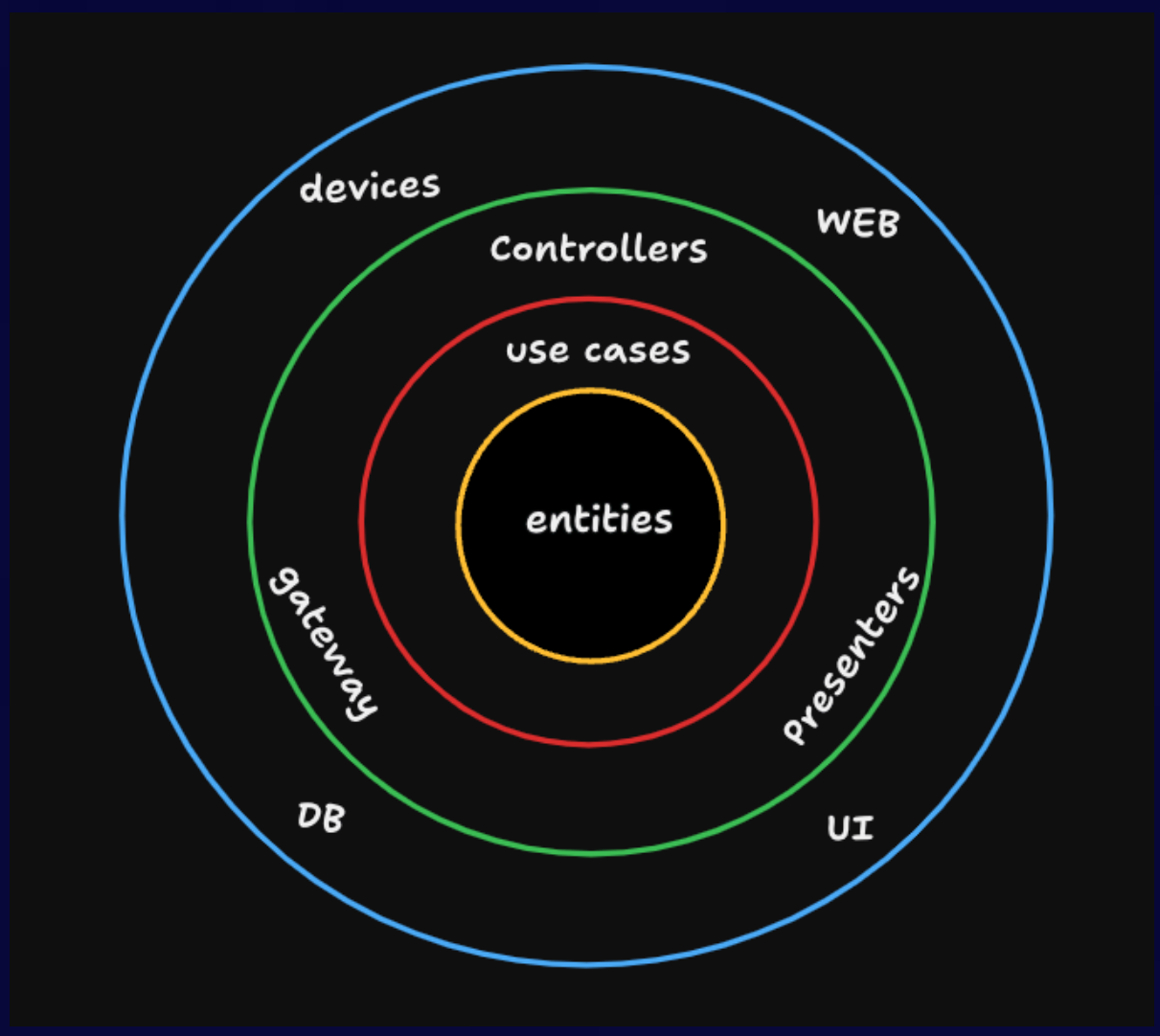
Desvendando
Clean Architecture



Isaac Gomes

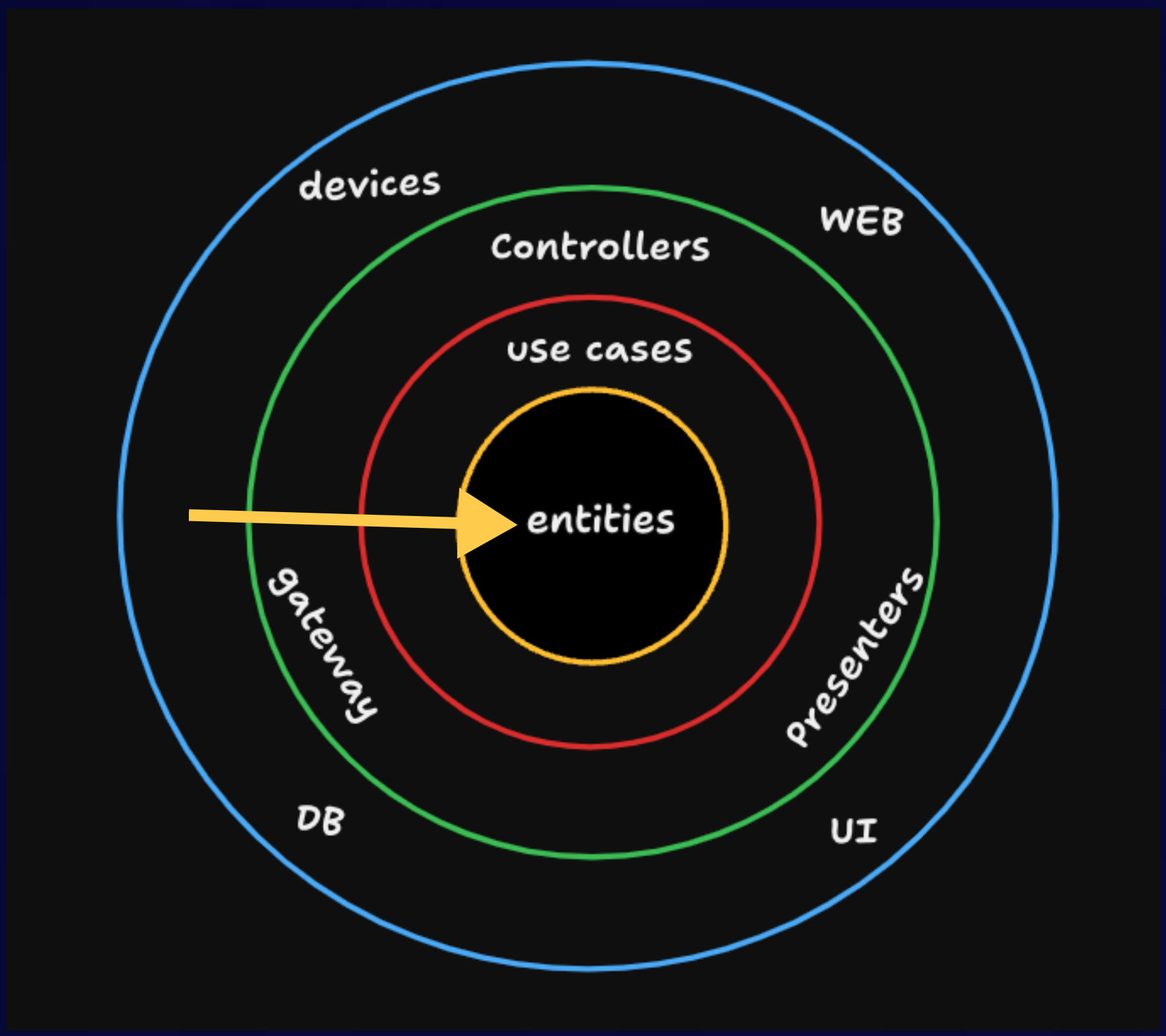
o que é Clean Architecture ?



Clean Architecture é um padrão de design de software que organiza o código em camadas independentes, facilitando a manutenção e escalabilidade. Ele separa a lógica de negócios da infraestrutura e da interface, promovendo uma estrutura modular e testável.

Isaac Gomes

o que é Clean Architecture ?



o fluxo de consumo de baseia de **fora para dentro** de uma maneira que temos uma **independência de camadas**

Isaac Gomes

o que é Entities ?

- Contém as **regras de negócio** mais genéricas e independentes

- Não depende de nenhuma outra camada.

Isaac Gomes

o que é useCases ?

- Contém a **lógica específica** da aplicação, ou seja, os fluxos de trabalho do sistema.

-São os serviços que coordenam as interações entre as **entidades**.

-Depende apenas das **entidades**.

Isaac Gomes

o que é Controllers ?

-Contém os **adaptadores** e controladores que convertem dados entre o formato interno das camadas Use **Cases/Entities** e o formato externo da camada **Frameworks & Drivers**.

-Exemplos incluem **controladores de API**, **repositórios**, e **transformadores de dados**.

Isaac Gomes

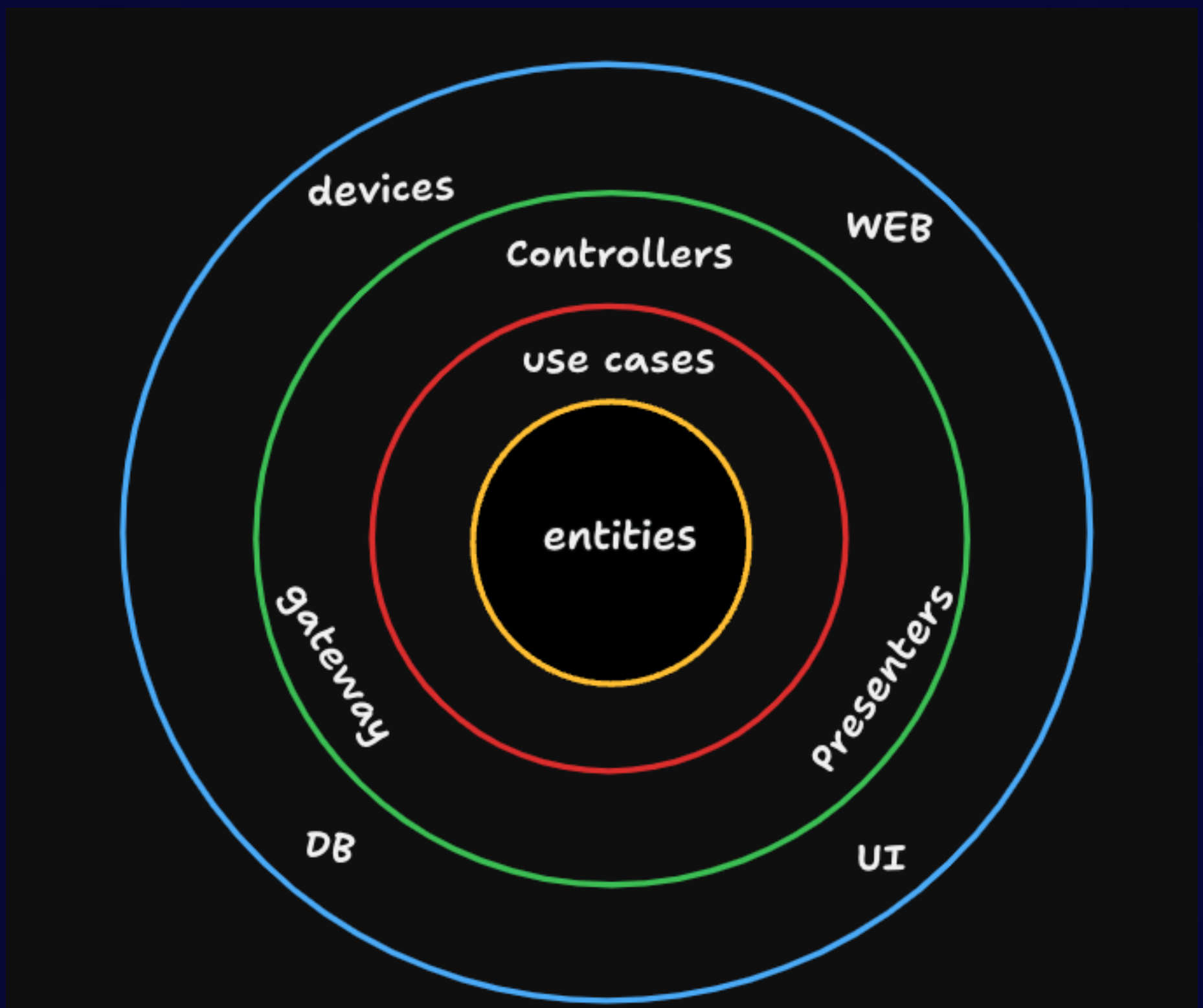
o que é Frameworks?

-Contém a implementação de detalhes como banco de dados, frameworks da web, etc.

-Esta camada é a mais externa e pode ser facilmente trocada.

Isaac Gomes

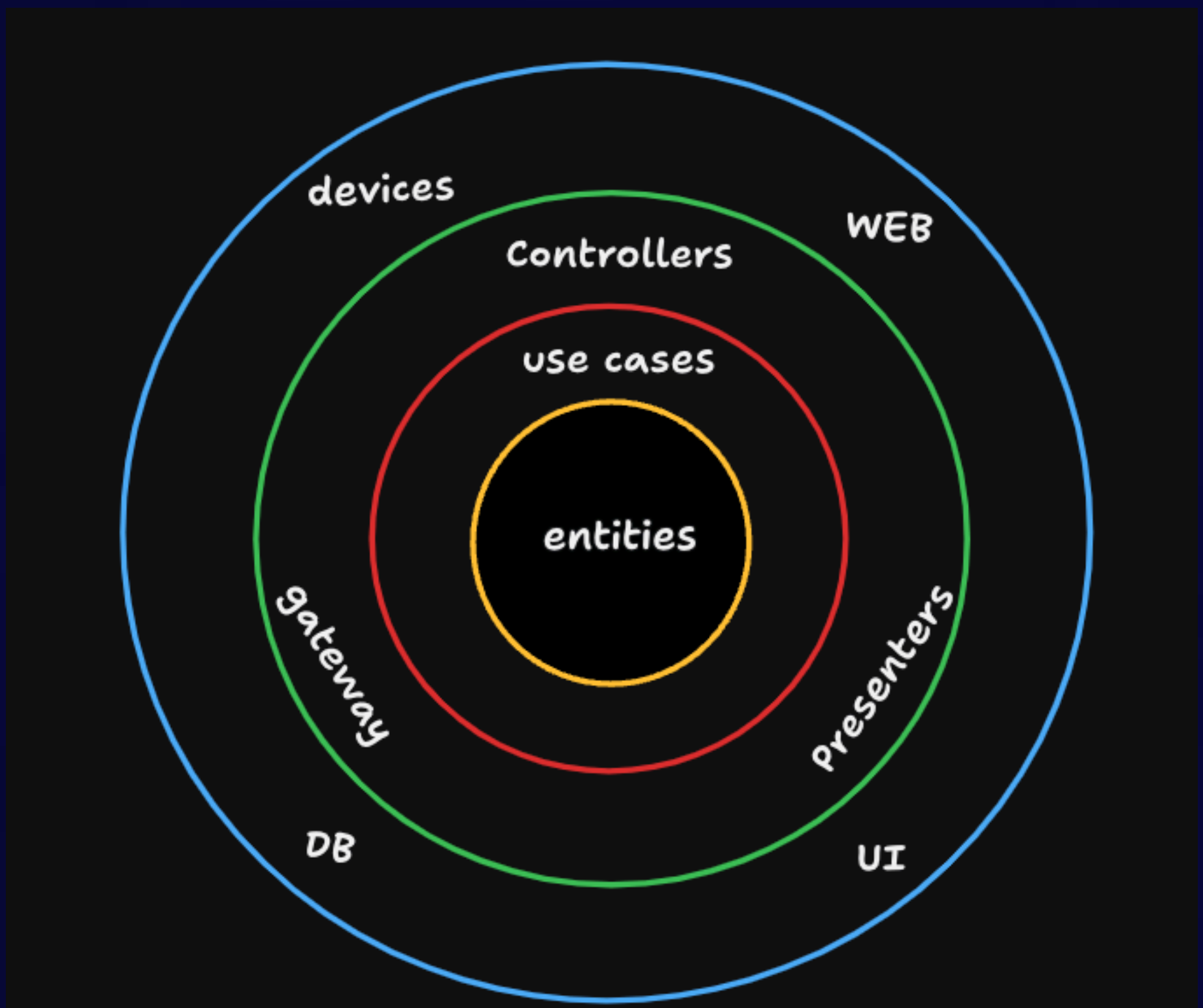
agora que sabemos o que cada camada representa



como fica a implementação disso na pratica?

Isaac Gomes

agora que sabemos o que cada camada representa



como fica a implementação disso na pratica?

Isaac Gomes

cenário

devemos criar **dois recursos**
de uma lista de tarefa

cadastrar uma nova tarefa
Listar todas as tarefas

Isaac Gomes

1- criar nossa entidade da Task

```ts
export class Task implements ITask {
  public id: string
  public title: string
  public description: string

  constructor({ id, title, description }: ITask) {
    this.id = id
    this.title = title
    this.description = description
  }
}
```

Isaac Gomes

TS

2 - criar nosso useCase

```
export class CreateTask {
  constructor(private taskRepository: TaskRepository) {}

  execute(title: string, description: string): Task {
    const task = new Task({
      id: generateId(),
      title,
      description,
    })
    this.taskRepository.save(task)
    return task
  }
}
```

Isaac Gomes

3 - vamos criar nosso TaskRepository

```ts
export class InMemoryTaskRepository implements TaskRepository {
  private tasks: Task[] = []

  save(task: Task): void {
    this.tasks.push(task)
  }

  findAll(): Task[] {
    return this.tasks
  }
}
```

Isaac Gomes

4 - vamos criar nosso TaskController

```
export class TaskController {
  private createTask: CreateTask
  private taskRepository: TaskRepository

  constructor(dependencies: TaskControllerDependencies) {
    this.createTask = dependencies.createTask
    this.taskRepository = dependencies.taskRepository
  }

  create(req: Request, res: Response) {
    const { title, description } = req.body
    const task = this.createTask.execute(title, description)
    res.json(task)
  }

  findAll(_req: Request, res: Response) {
    const tasks = this.taskRepository.findAll()
    res.json(tasks)
  }
}
```

Isaac Gomes

TS

5 - agora vamos expor isso no nosso framework

```ts
const app = express()
app.use(express.json())

const taskRepository = new InMemoryTaskRepository()
const createTask = new CreateTask(taskRepository)
const taskController = new TaskController({
  createTask,
  taskRepository,
})

app.post('/tasks', (req, res) => taskController.create(req, res))
app.get('/tasks', (req, res) => taskController.findAll(req, res))

app.listen(3000, () => console.log('Server is running on port 3000'))
```

Isaac Gomes

6 - agora é só testar {create}

POST http://localhost:3000/tasks

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
  "title": "Sample Task",
  "description": "This is a description of the sample task."
}
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "id": "ud9ggdcd35",
    "title": "Sample Task",
    "description": "This is a description of the sample task."
}
```

Isaac Gomes

TS

6 - agora é só testar {getAll}

GET http://localhost:3000/tasks

Params Authorization Headers (7) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1
```

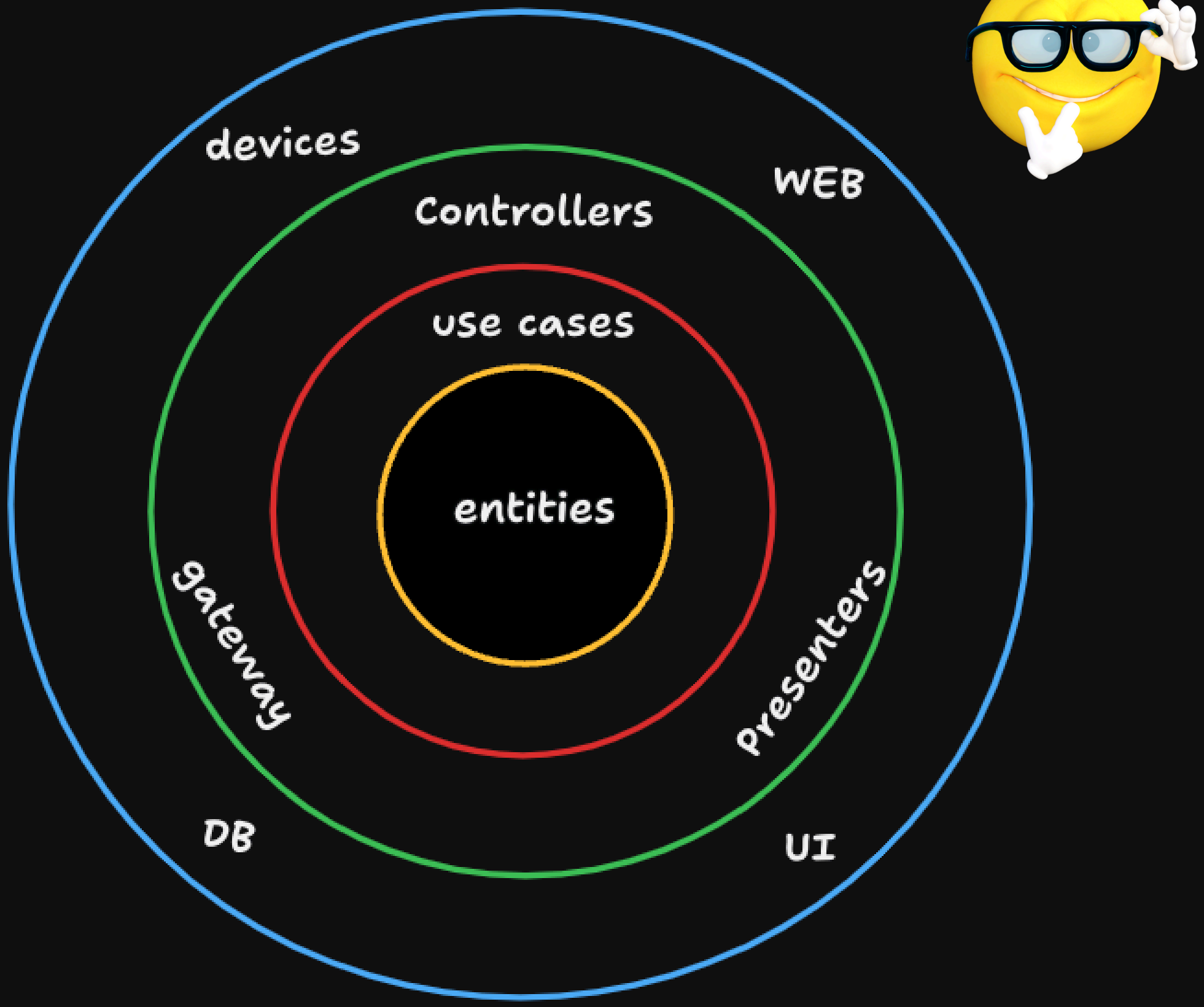
Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
[
    {
        "id": "ud9ggdcd35",
        "title": "Sample Task",
        "description": "This is a description of the sample task."
    }
]
```

Isaac Gomes

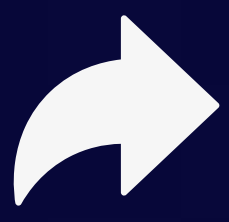
agora as coisas começam a tomar forma



Isaac Gomes

Gostou?

Curta



Compartilhe

Salve

Isaac Gomes