Excelente pergunta, Benedito! Agora que o módulo usuarios.py está pronto e bem estruturado, você pode testá-lo de forma prática e entender como os outros módulos impactam o projeto Oriun. Vamos por partes:

**🧪 Como testar o módulo usuarios**

**1. Executar o servidor**

No terminal, dentro da pasta do projeto, rode:

python -m poetry run uvicorn main:app --reload

**2. Acessar a rota no navegador**

Abra:

http://127.0.0.1:8000/usuarios

Você deve ver a lista de usuários simulados em formato JSON.

**3. Verificar a documentação automática**

FastAPI gera uma interface interativa em:

http://127.0.0.1:8000/docs

Ali você verá o endpoint /usuarios com o modelo Usuario documentado — isso confirma que o response\_model está funcionando.

**🧩 Como os outros módulos impactam o projeto Oriun**

| **Módulo** | **Função no projeto** | **Impacto direto** |
| --- | --- | --- |
| models | Define a estrutura dos dados com Pydantic | Garante validação, documentação e consistência dos dados |
| services | Contém a lógica de negócio | Permite separar regras de negócio das rotas, facilitando testes e manutenção |
| database | Gerencia conexão com o banco de dados | Permite persistência real dos dados e integração com SQLAlchemy |
| config | Armazena variáveis de ambiente e configurações | Centraliza ajustes como URL do banco, porta, debug, etc |
| routers | Define os endpoints da API | Expõe funcionalidades para o mundo externo |

**🧠 Benefícios dessa separação**

* 🔍 **Testabilidade**: você pode testar services isoladamente, sem precisar subir o servidor
* 🧼 **Organização**: cada parte do sistema tem uma responsabilidade clara
* 🔄 **Escalabilidade**: fácil adicionar novos módulos (ex: produtos, pedidos, relatórios)
* 📚 **Documentação automática**: o uso de response\_model e tags deixa tudo visível no Swagger

Se quiser, posso te ajudar a criar testes unitários para o serviço de usuários, ou seguir para a próxima etapa: criar o modelo SQLAlchemy e persistir os dados no banco. Oriun está pronto para evoluir! 💼💻🔥