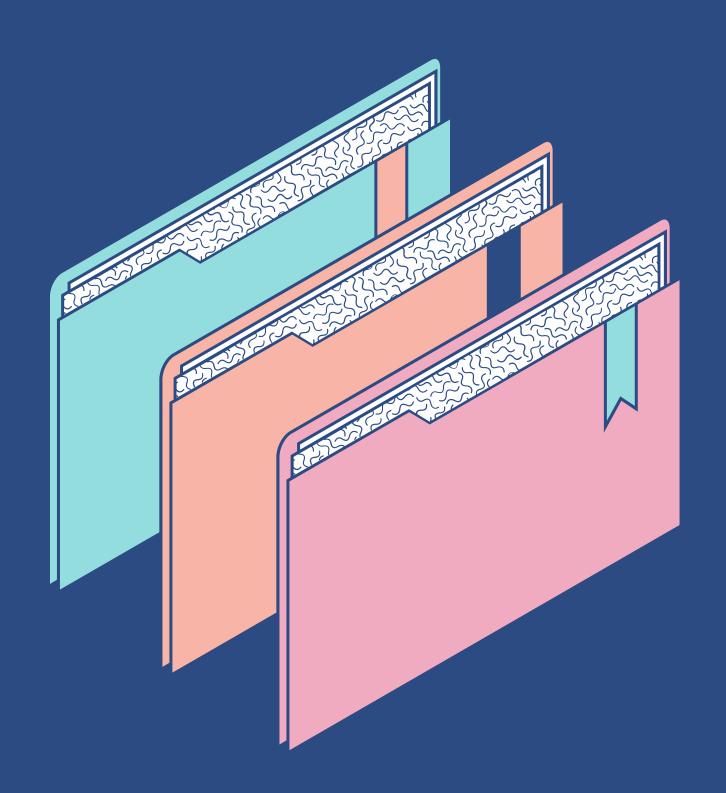


# Arquitectura de Desarrollo de software DDD (Domain-Driven Design)

### **Estudiantes:**

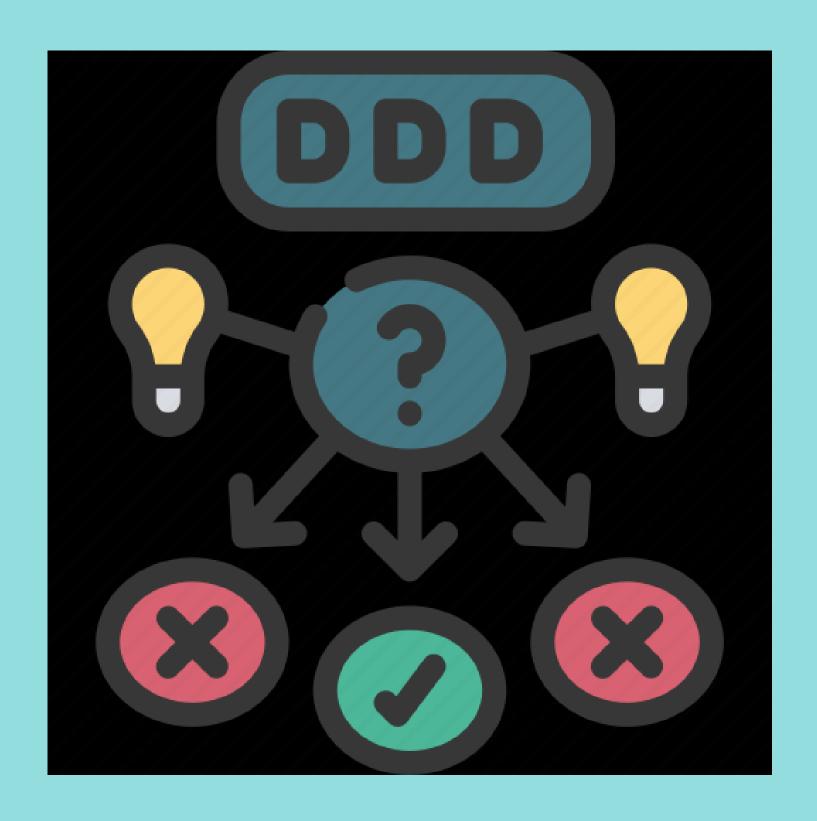
- ·Rene Manuel Avilés Montecinos.
- ·José Milton Mamani Callizaya.
- ·Roberto Carlos Aruquipa Arzala.
- ·Diego Carlos Rojas Apaza.

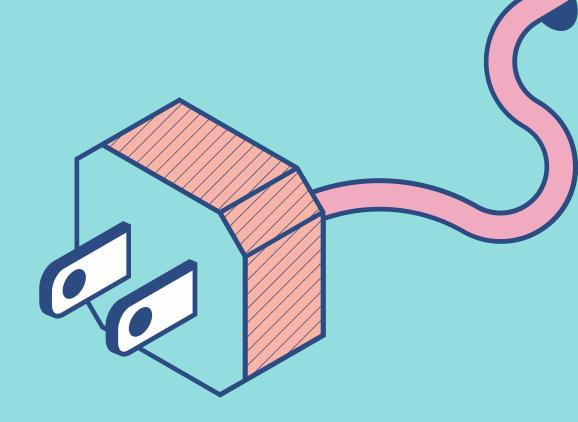


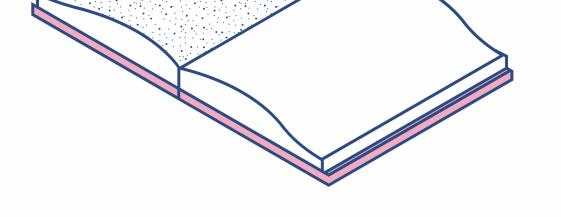
# Índice

- Introducción
- Conceptos
- Desafíos y Consideraciones
- Objetivos
- Marco Teórico
- Marco Práctico
- Conclusiones

## Introducción

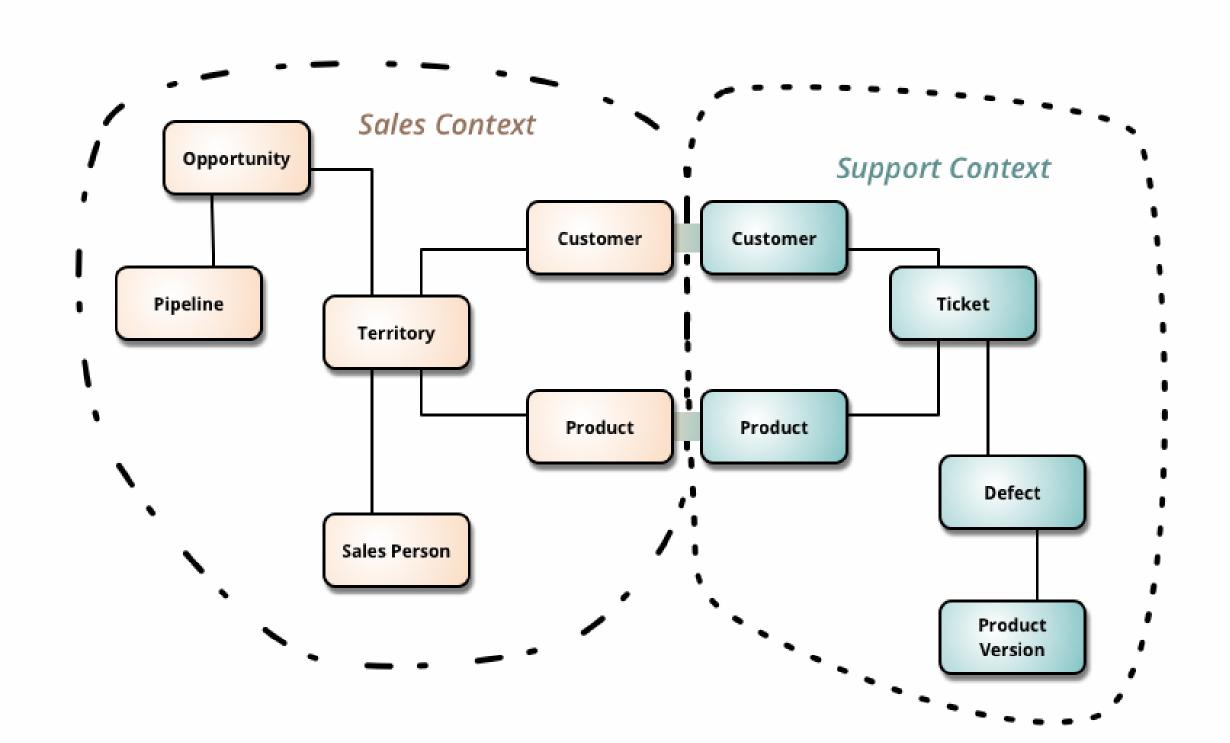






### Conceptos

- Modelo de Dominio
- Ubicuidad del Lenguaje
- Bounded Contexts
- Objetos de valor



## Desafíos y Consideraciones

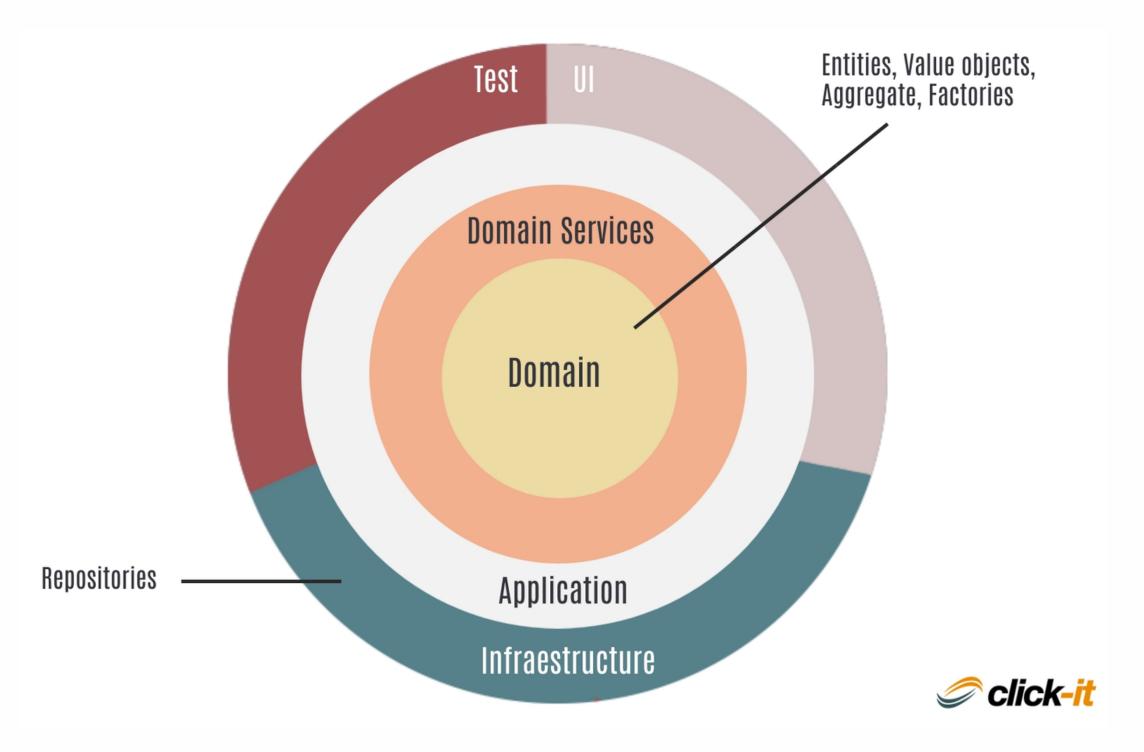
- Complejidad del Dominio
- Expertos en la materia
- Colaboración del Equipo
- Comunicación
- Cambios Constantes



# Objetivos

- -Comprender el dominio
- -Crear un modelo de dominio
- -Perfeccionamiento
- -Implementación del modelo dominio





### Marco Teórico

- Proceso de diseño con DDD
- Implementación de DDD
- Patrones y Buenas Prácticas de DDD
- Integración con otros Aspectos del Desarrollo
- Herramientas y tecnologías

# Marco Práctico

Programa de ejemplo:

"Reserva de mesas para restaurants"

Con NestJS

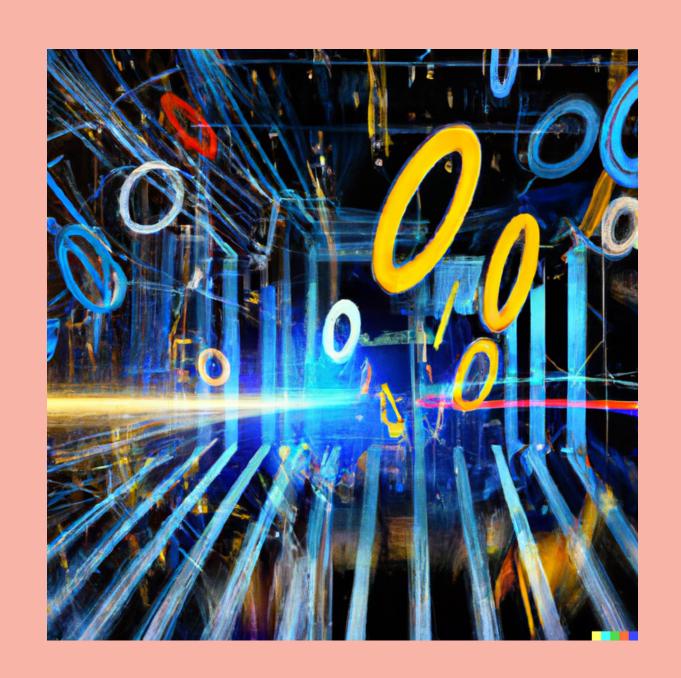
```
reservations.controller.spec.ts U X
                                                     src > reservations > controllers > 15 reservations.controller.spec.ts > ...
                                                            import { Test, TestingModule } from '@nestjs/testing';
 X rs reservations.controller.spec.ts src\reservations\co... U
                                                            import { ReservationsController } from './reservations.controller';
                                                           describe('ReservationsController', () => {
                                                             let controller: ReservationsController;
                                                             beforeEach(async () => {
                                                              const module: TestingModule = await Test.createTestingModule({
                                                                 controllers: [ReservationsController],
                                                               }).compile();
   reservations.controller.spec.ts
                                                               controller = module.get<ReservationsController>(ReservationsController);
                                                              it('should be defined', () => {
                                                              expect(controller).toBeDefined();
 app.controller.spec.ts
.gitignore
nest-cli.json

  package-lock.json

    package.ison

₩ README.md
tsconfig.build.json
```

### Conclusiones





El DDD puede ayudar a crear sistemas de software que reflejen con precisión el dominio del problema y sean fáciles de entender, mantener y evolucionar con el tiempo.



# iGRACIAS!

