Calidad del producto software

CI / CD

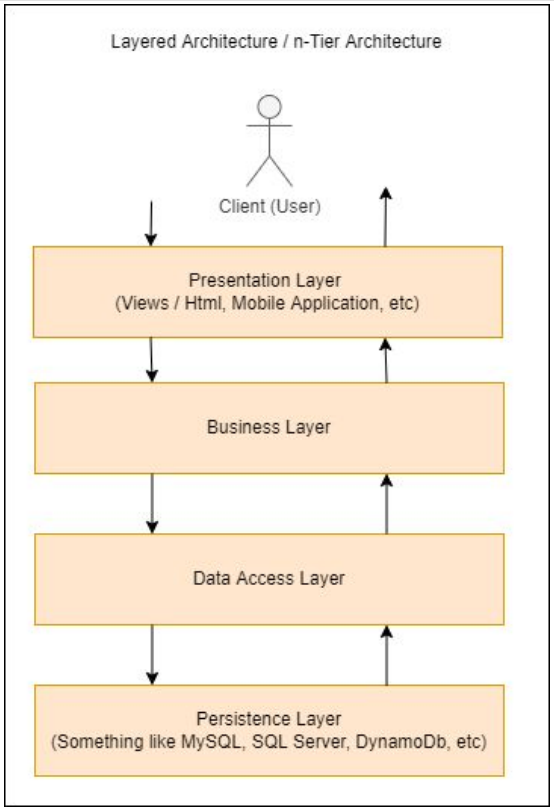
[Continuous Integration / Continuous Deployment]

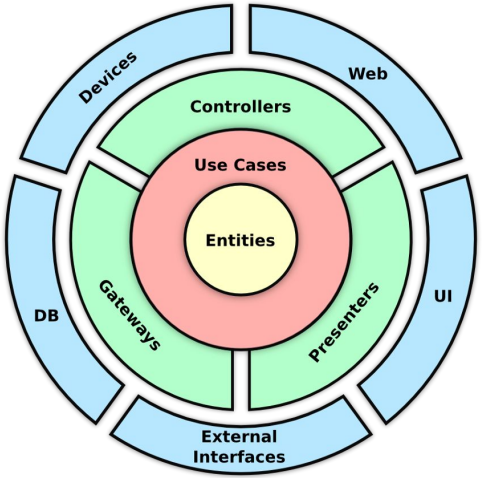
- ¿Qué herramienta de integración continua usan y cómo está configurada?
- ¿Por qué eligieron esa herramienta?
- ¿Quién la mantiene?

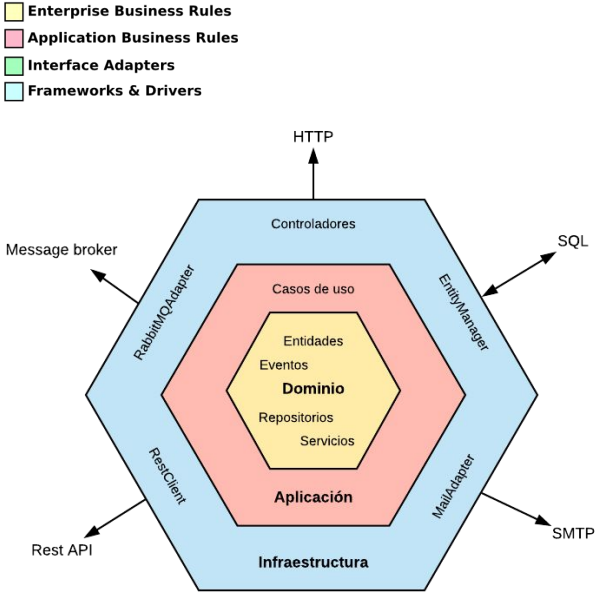
- ¿Dónde van a desplegar el producto?
- ¿Cuál es el mecanismo para desplegar una nueva versión?
- ¿Cómo manejan las configuraciones ambientales?

Fly.io

Arquitectura interna del producto

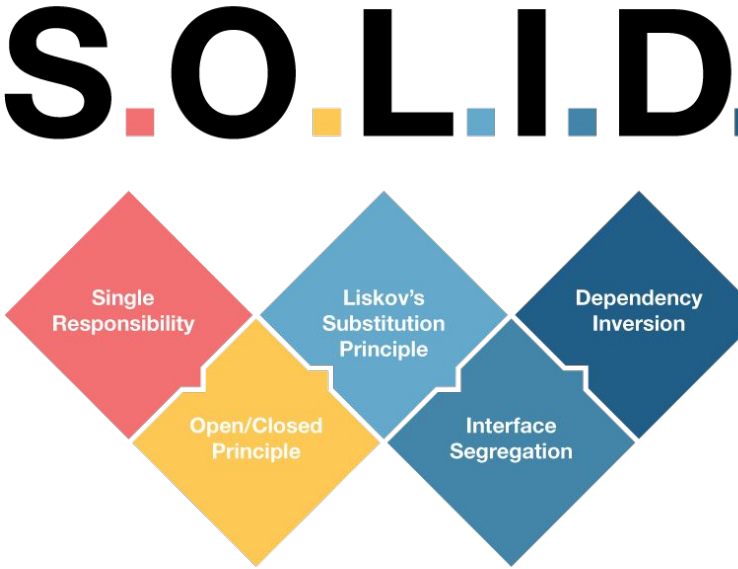






Calidad de código: principios SOLID

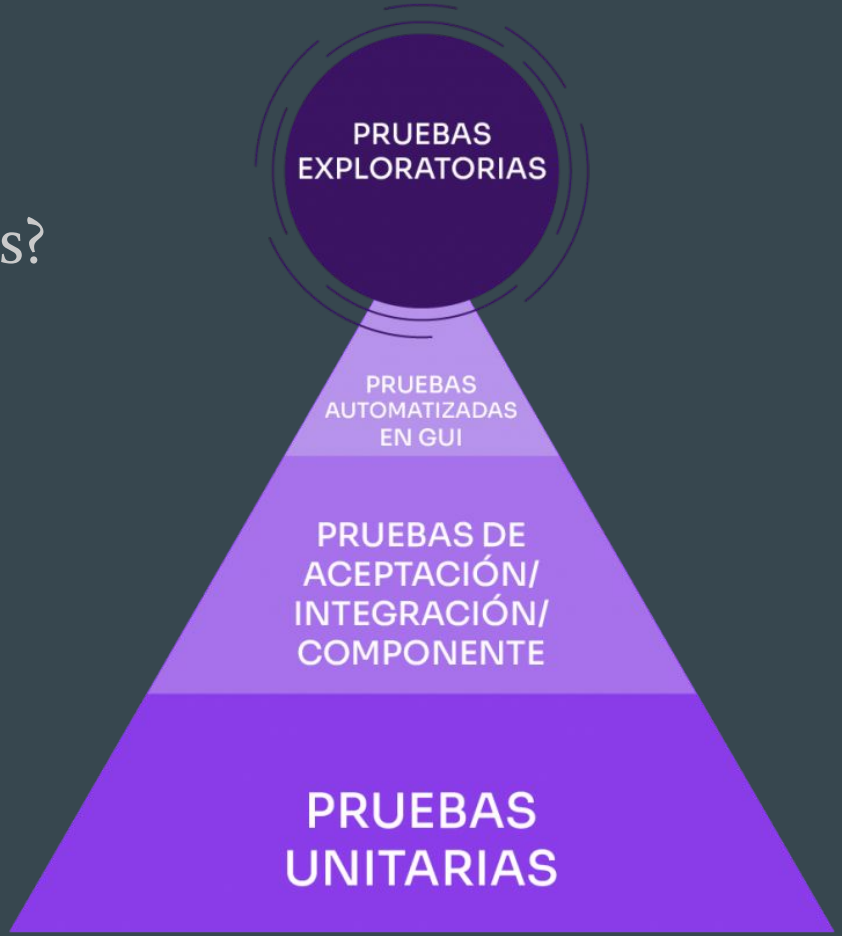
- Single responsibility
  - ¿Cómo está organizado el código?
  - Cada modulo hace una unica cosa?
- **¿Cómo son las dependencias entre los módulos?**
- **¿Aplican el principio de Inversión de Dependencias*?**



*
https://stackify.com/dependency-inversion-principle/
https://medium.com/javarevisited/how-dependency-inversion-and-dependency-injection-works-db48b09ffa17

Pruebas

- ¿Cuál es la estrategia de pruebas?
- Automatización
  - **Unit test** ✓
  - Integración ?
  - Aceptación ?
  - Interfaces externas ?



Pruebas automatizadas

- ¿Cómo están hechas las pruebas?
- ¿Qué frameworks de testing usan?
- ¿Qué herramienta de mocking usan?
- ¿Qué semántica usan en los tests?
  - Given When Then?
  - Nombres de los tests: “xxxxxxxTest” / “xxxxxxxShould” ?

```
Scenario Outline: Only correctly structured emails should be accepted
  Given Candy does not have a Frequent Flyer account
  When Candy tries to register with an email of "<email>"
  Then she should be told "Not a valid email format"
```

Seguridad

- ¿Qué mecanismo usan para manejar accesos, roles, etc?
- ¿Cómo manejan los “secretos” del producto?
  - Dónde
  - Cómo
- https

Infraestructura

- Persistencia
  - ¿Dónde persisten los datos?
  - ¿Por qué eligieron ese **tipo** de persistencia y esa **herramienta** de persistencia?
- Virtualización
- APIs externas
  - ¿Para que y por que?
  - ¿Qué alternativas manejaron antes de decidir?
  - ¿Cómo prueban los componentes que interactúan con la API *SIN* integrarse realmente con ella?

Prácticas técnicas

- Control de versiones
  - Usan feature branches, trunk based development, etc?
  - ¿Por qué esa elección y como la implementan en el día a día?
  - ¿Cómo le funciona al equipo?
- ¿Hacen TDD?
- ¿Hacen Pair Programming?
- Collective Code Ownership
  - Se dividen el trabajo? (algunos front y otros back?)
  - Todos deben conocer y tocar TODO el código

Clean Code

- Idioma
- Nombres (módulos, clases, funciones, variables)
  - Son coherentes?
  - Tienen homogeneidad conceptual?
  - Son claros?
  - Se usa algun standard?
- Orden de funciones: +arriba = -detalle
- Las funciones son cortas?
  - Regla de las 3 líneas
- Hay IFs, Switch, etc?

Forma de trabajo

- 3 encuentros virtuales de una hora aprox. (justo antes de cada MVP) con el equipo COMPLETO
- Revisión con el grupo de una serie de cuestiones técnicas predefinidas
- Discusión en conjunto sobre decisiones tomadas y alternativas de implementación
- Listado de puntos de mejora a ser revisado en el siguiente encuentro
- Se deben cumplir una serie de criterios base para la aprobación
- Nota sobre la calidad técnica del producto para el equipo que es considerada junto con el resto de las evaluaciones


That's all Folks!