10 Pines

Diseño a la Gorra - Episodio 08
Buenos Aires vs. (London vs. Chicago) - Parte 1

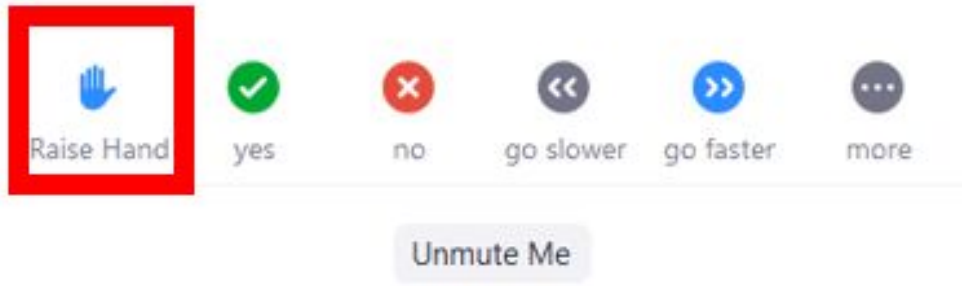
Hernán Wilkinson

hernan.wilkinson@10pines.com

@HernanWilkinson

https://alagorra.10pines.com

Estar muteados a menos que sea necesario



No voy a poder leer el chat

La comunicación visual es importante. Usarla a discreción

Diseño ¡a la gorra!

¡Bienvenidos!
Durante esta serie de Webinars exploraremos *qué* significa **Diseñar Software con Objetos** y *cómo* lo podemos hacer cada vez mejor.
Trataremos muchos temas que irán desde cuestiones filosóficas como qué significa Diseñar en nuestra profesión y dónde está expresado ese Diseño, pasando por consejos y heurísticas para diseñar "mejor" y terminado con ejemplos concretos de cómo aplicar esas heurísticas en la *vida real.*
Los webinars son "*language agnostic*", o sea que no dependen de un lenguaje de programación en particular, aunque los ejemplos que usaremos estarán hechos principalmente en **Java**, **JavaScript**, **Ruby**, **Python** y mi querido **Smalltalk** cuando amerite 😉 .

Te esperamos todos los Martes a las 19 Hrs GMT-3 a partir del Martes 11 de Agosto de 2020.
Para poder participar tenes que registrate acá.

Todo el código y presentaciones estarán disponibles para que lo puedan usar y consultar en cualquier momento acá.

¡Trae ganas de aprender y pasarla bien!

¿Por qué a la Gorra?
Al igual que cuando Diseñamos Software está bueno usar una **Metáfora** para entender qué estamos modelando, en este caso usamos una metáfora para explicar cómo *financiaremos*

Donaciones

mercado pago

$100 - Casi una 🍷 Pagar

$250 - Una buena 🍺 Pagar

$500 - Menos que 🍔 +🍟 Pagar

PayPal Donate

¿Querés donar un monto variado o en otra plataforma?

Dictado por:
Hernán Wilkinson

https://alagorra.10pines.com

Online

Diseño Avanzado de Software con Objetos I

Empieza el **13/10**

Del 13/10 al 22/10. Días: 6 días — Del 13 al 16, 20 y 22 de Octubre — 9:00 a 13:00 GMT-3.

AR$ **16.200-** (IVA incluído)

Hacer una consulta

Inscribirme ahora 10% Dto.

Dictado por:
Máximo Prieto

~~AR$ 18.000-~~

AR$ 16.200-

(IVA incluído)

~~U$D 300-~~

U$D270-

(impuestos incluídos)

~~Super Early Bird~~

Early Bird

https://academia.10pines.com/courses/96-diseno-avanzado-de-software-con-objetos-i

Online

Test Driven Development Avanzado

Empieza el **02/11**

AR$ **12.750-** (IVA incluído)

Hacer una consulta

Inscribirme ahora 15% Dto.

Del 02/11 al 06/11. Días: Lunes a Viernes — 9:00 a 13:00 GMT-3.

Dictado por:
Hernán Wilkinson

~~AR$ 15.000-~~

AR$ **12.750-**

(IVA incluído)

~~U$D 250-~~

U$D215-

(impuestos incluídos)

Super Early Bird

~~Early Bird~~

https://academia.10pines.com/courses/95-test-driven-development-avanzado

¿Qué vimos en el último episodio?

Categorías

- Tipos de Tests
- Estructura de los Tests
- Datos de Prueba vs. Casos de Prueba
- “Inducción Incorrecta”
- Cómo nombrar Tests
- El Test tiene que estar en “control de todo”
- Reificación de Aserciones
- Testing de Excepciones

Buenos Aires vs. (London vs. Chicago)

Words of Caution

El nombre “***Buenos Aires***” es solo por una cuestión de marketing

No puedo arrogarme el hecho de que “todo Buenos Aires” programe cómo lo voy a hacer yo


LONDON
Meets
CHICAGO


Speculative development
complexity caused by design
Two

Descripción del Problema

- Registrar un nuevo usuario
- Logearse
- Crear “posts” (publicaciones)
  - No se puede postear con palabras “inapropiadas” como elephant, ice cream y orange
- Obtener los posts de un usuario (timeline)
- Seguir a otro usuario
- Obtener los posts propios y de los que sigo (wall)
- Obtener todos los usuarios
- Obtener los usuarios seguidos por uno en particular

Ejemplo: Registrar nuevo usuario

POST `/users` Register a new user.

Parameters — Try it out

Name	Description
body * required `object` *(body)*	New user information **Example Value** | Model `{ "username": "string", "password": "string", "about": "string" }` **Parameter content type** application/json

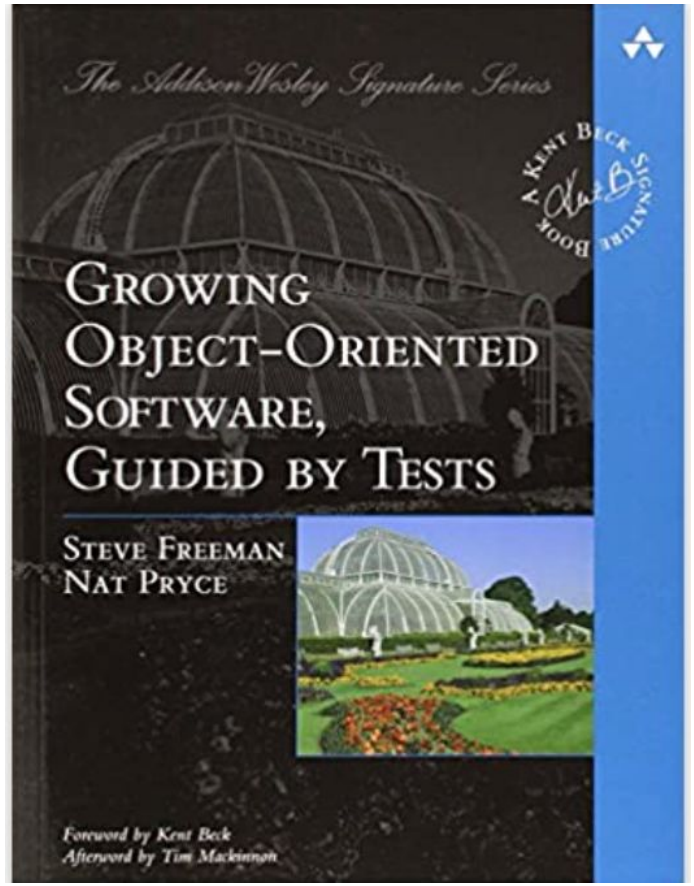
Responses — Response content type: application/json

Code	Description
201	User created. **Example Value** | Model

El ejercicio viene con

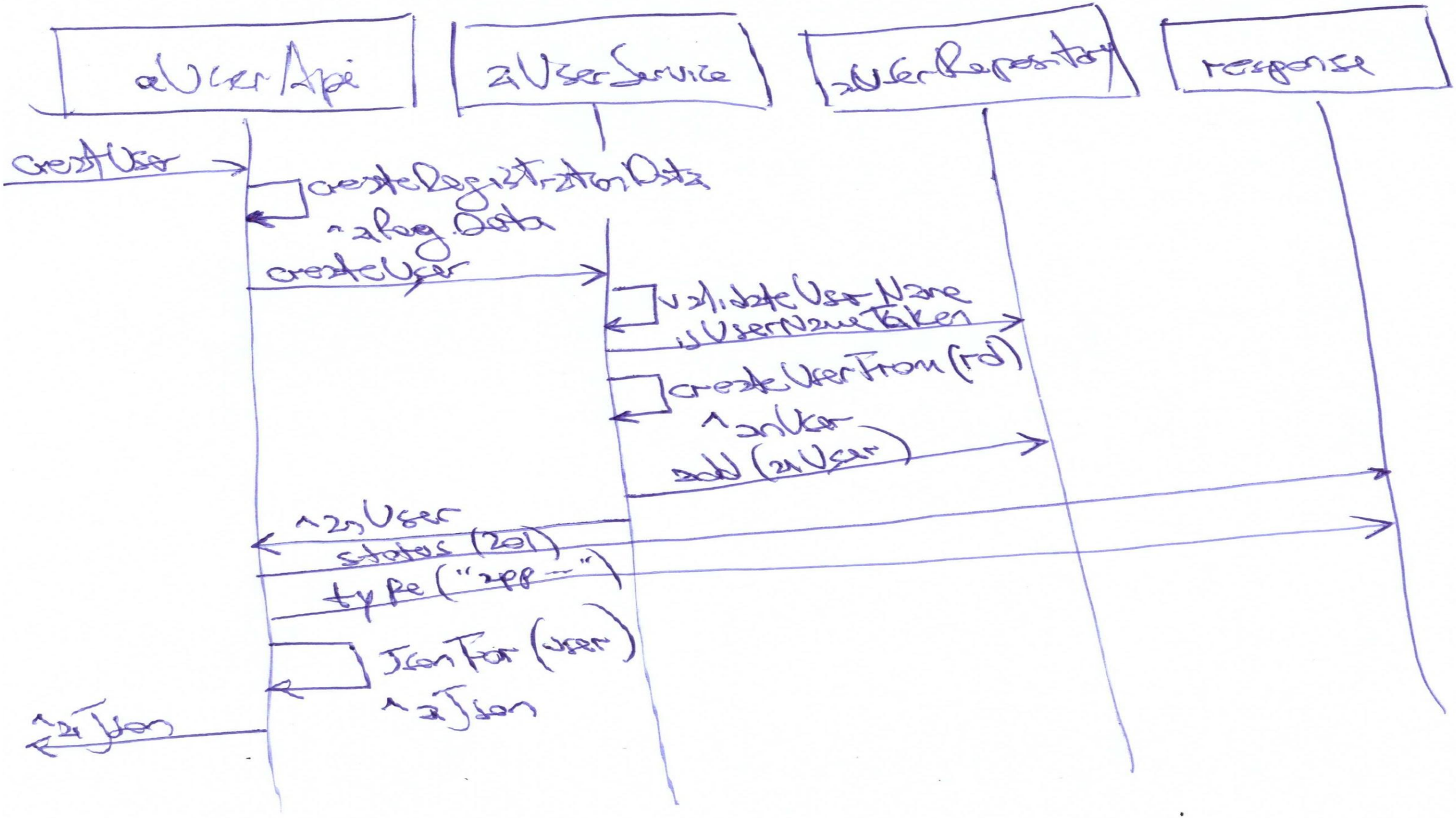
- Descripción de la API
- Tests de Aceptación
- User Interface (No provista pero usada en los videos)

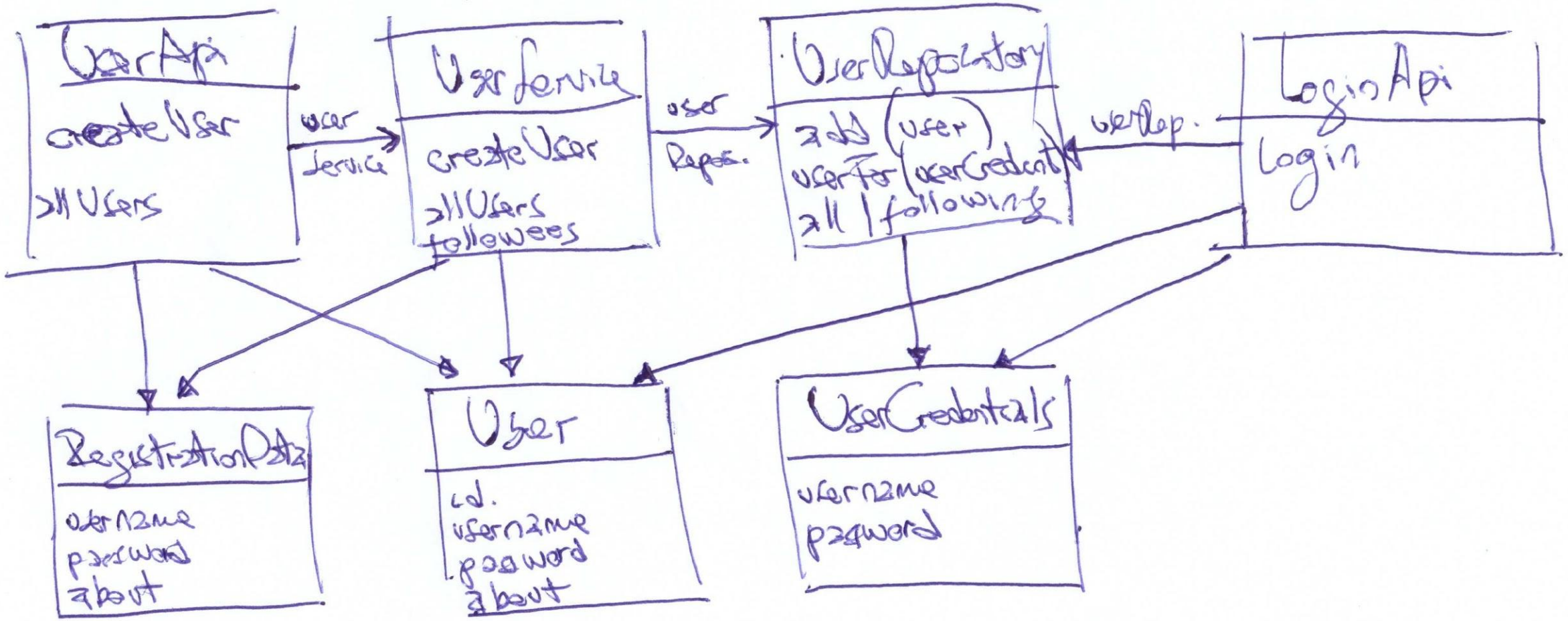
London Way - Sandro Mancuso



- “Test Unitarios” que mockean todo lo que no se testea directamente
- Se empieza desde el tope del árbol de ejecución (interfaces)
- Los tests definen cómo debe ser la implementación→ hay que tener la idea de cómo implementarlo de entrada

Veamos el Código


aUserApi
zUserService
xUserRepository
response
createUser
createRegistrationData
^zReg.Data
createUser
vxValidateUserName
isUserNameTaken
createUserFrom(rd)
^xnUser
add(zUser)
^z,User
status(201)
type("app...")
JsonFor(user)
^zJson
^zJson


UserApi
createUser
>|| Users
user
UserService
createUser
>|| Users
followees
user
Repos.
UserRepository
add (user)
userFor (userCredent
all | following
userRep.
LoginApi
login
RegistrationPojo
username
password
about
User
id.
username
password
about
UserCredentials
username
password

Conclusiones - Diseño

- Diseño orientado al problema computable, no al negocio
- Organización del proyecto en base al problema computable
- Uso de DTOs para comunicación entre “capas”
- Sobre uso de Json (a mi parecer)
- Uso de ids en vez de objetos
- 5 clases para registrar un usuario…
- Se expone la password a objetos externos

Conclusiones - Tests

- Tests de APIs de caja blanca
  - Opinión personal: Solo los podes hacer así si ya sabes cómo vas a hacer la implementación
- Los tests siguen la misma estructura
- Hay casos sin testear
  - Usuario con username vacío
  - Se puede usar followers y followees que no existen
  - Podés seguirte a vos mismo
  - etc.

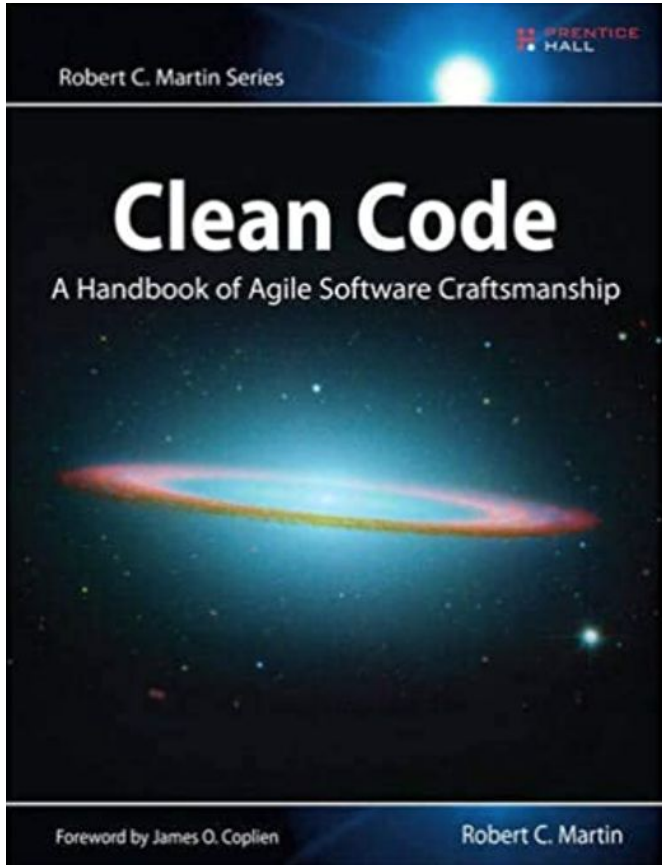
Frases

- "I don’t like to use classes that are not mine"
- "We are using the mocking to design the behavior"
  - Ya conoce lo que quiere hacer
- "Mocking is not about testing, it is about designing the collaboration between classes"
  - Las clases no colaboran. Los objetos si
- “Mocking tests are couple to implementation” (Episodio 1 - 1:13:25)

Hechos

- 11 minutos en escribir el primer test
  - Clases: UserAPI, UserService, RegistrationData, User
  - 4 Mocks con varias configuraciones
- 4 minutos para hacerlo pasar

Chicago Way - Bob Martin



- Crear clases que representan Casos de Usos
  - CreateUser, Login, GetUsers, etc.
- Escribir tests en base a las clases de caso de uso
- No empezar por las interfaces
- No usar mocking
- Estilo “más procedural” de solución

Veamos el Código

Conclusiones - Diseño

- Clases que representan Casos de usos
- Una clase por API
- UseCaseContext, APIContext → Objetos globales
- DTOs para comunicación entre “capas”
  - CreateUserRequest igual a User
- Uso de variables de instancia públicas
- Clases sin comportamiento (estructuras de datos)
- Clases con un solo método que únicamente forwardean mensaje

Conclusiones - Diseño

- Inconsistencia en el manejo de casos de error
  - A veces levanta excepción (Ej: CreateUser)
  - A veces retorna boolean (Ej: Subscribe)
- Copia de objetos debido a no ser inmutables
- Se expone la password a objetos externos, que además es pública
- No hay clases “service”

Conclusiones - Tests

- Tests de Excepciones sin aserciones de que no sucedió lo que no tenía que pasar
  - throwsExceptionIfDuplicateIsCreated
- setUp muy grande en mi opinión
- Se puede postear con palabra inapropiadas

Conclusiones - Tests

- Hay casos sin testear
  - Usuario con username vacío
  - Si follower o followee no existe, lo relaciona con null
  - Podés seguirte a vos mismo
  - etc.

Frases

- "We have tested that the assignment operator works" (Episodio 6 - 19:45)
- “I use data structure without constructors because they tend to grow”

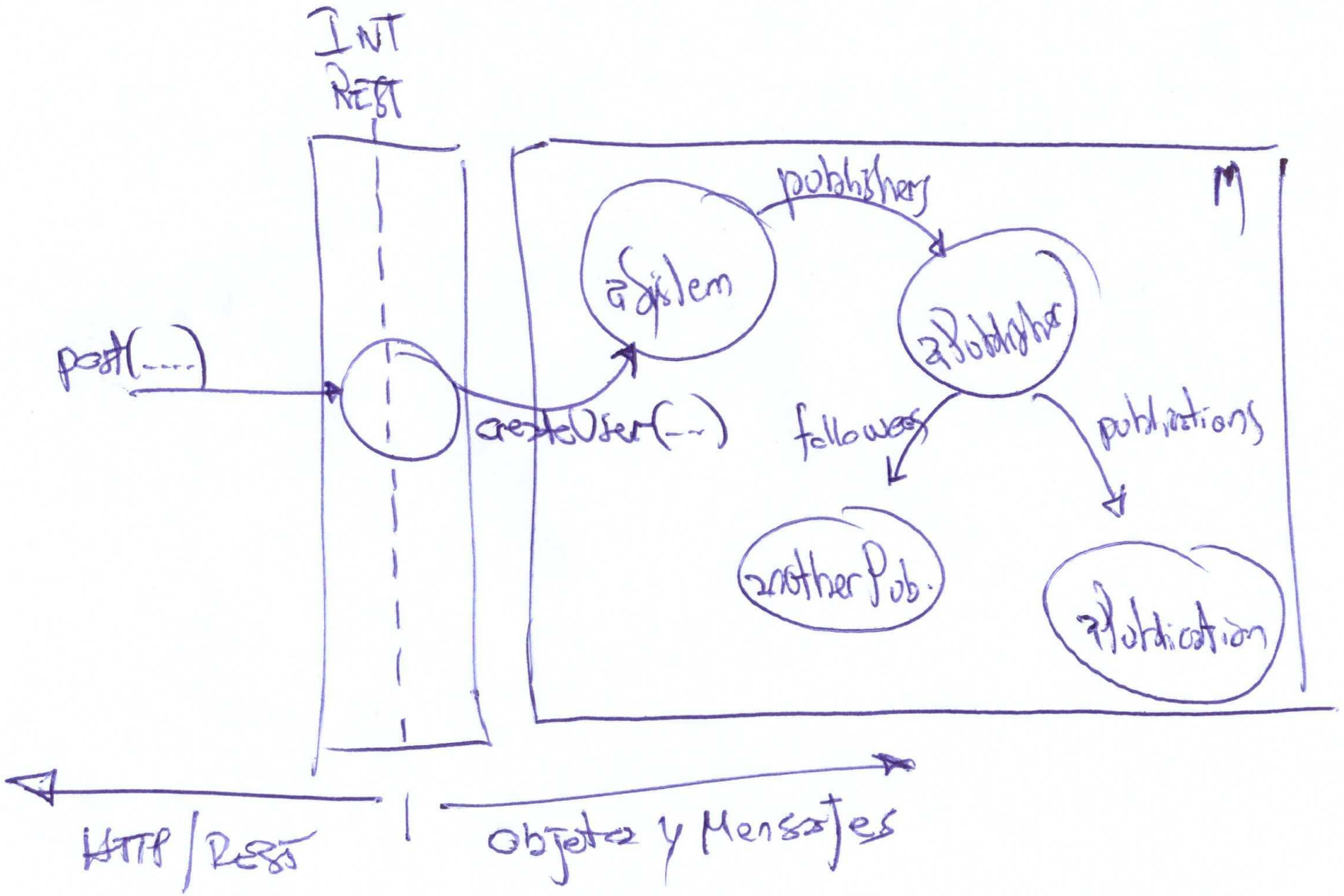
Hechos

- La primera vez que corre tests, son 2 tests juntos y pasan de entrada (Episodio 6 - 14:15)

Buenos Aires way

- Empezar testeando y modelando el negocio
- Dejar Interfaces para el final
- Usar metáfora para concretizar el negocio
- Diseñar objetos con heurísticas vistas hasta ahora

Ejemplo de Arquitectura


INT
REST
post(...)
createUser(...)
publishers
followers
publications
HTTP / Rest
Objetos y Mensajes

¡A programar!

Comparaciones

	London	Chicago	Buenos Aires	BsAs/London	BsAs/Chicago
Clases de Modelo	16	15	6	38%	40%
Lineas de Código de Modelo	353	275	248	70%	90%
Clases de Tests	8	10	3	38%	30%
Lineas de Código de Test	344	322	271	79%	84%
Cantidad de tests	28	19	27	96%	142%
Tiempo de ejecucion de tests	154	153	110	71%	72%
Sin cubrir	26%	12%	0%		

Conclusiones

- Diseño basado en el negocio
- Se pudo resolver el problema sin mocks ni clases ficticias (use cases)
- Se usaron objetos completos, válidos, inmutables y sin null
- No se rompió el encapsulamiento
- No fue necesario usar DTOs, Json, etc.
- No fue necesario usar Ids entre los objetos de negocio
- Por concentrarnos en el negocio, detectamos más casos de error

¡Lo que viene!
¡Lo que viene!

Lo que viene

- Implementar la interfaz Rest
- Hacerlo persistente

Preguntas

Diseño ¡a la gorra!

¡Bienvenidos!
Durante esta serie de Webinars exploraremos *qué* significa **Diseñar Software con Objetos** y *cómo* lo podemos hacer cada vez mejor.
Trataremos muchos temas que irán desde cuestiones filosóficas como qué significa Diseñar en nuestra profesión y dónde está expresado ese Diseño, pasando por consejos y heurísticas para diseñar "mejor" y terminado con ejemplos concretos de cómo aplicar esas heurísticas en la *vida real.*
Los webinars son "*language agnostic*", o sea que no dependen de un lenguaje de programación en particular, aunque los ejemplos que usaremos estarán hechos principalmente en **Java**, **JavaScript**, **Ruby**, **Python** y mi querido **Smalltalk** cuando amerite 😉 .

Te esperamos todos los Martes a las 19 Hrs GMT-3 a partir del Martes 11 de Agosto de 2020.
Para poder participar tenes que registrate acá.

Todo el código y presentaciones estarán disponibles para que lo puedan usar y consultar en cualquier momento acá.

¡Trae ganas de aprender y pasarla bien!

¿Por qué a la Gorra?
Al igual que cuando Diseñamos Software está bueno usar una **Metáfora** para entender qué estamos modelando, en este caso usamos una metáfora para explicar cómo *financiaremos*

Donaciones

mercado pago

$100 - Casi una 🍷 Pagar

$250 - Una buena 🍺 Pagar

$500 - Menos que 🍔 +🍟 Pagar

PayPal Donate

¿Querés donar un monto variado o en otra plataforma?

Dictado por:
Hernán Wilkinson

https://alagorra.10pines.com

Muchas gracias

10 Pines

Creative Software Development

10pines.com

info@10pines.com

+54 (011) 6091-3125 / 4893-2057

Av. Leandro N. Alem 896 6° - Bs. As. - Argentina

@10pines