XKNATIS

ARCHITECTURE

Designer une application avec la Programmation Orientée Objet

XDNATIS

Compétence demandée :
Savoir construire le design d'une application web

XKNATIS

1. Rappel architecture MVC
2. Layered architecture
3. UML : Diagramme de classes
4. Exercices

XWIKI SAS

XWIKI NATIS

ARCHITECTURE D'UN APPLICATION WEB

XWIKI NATIS

Couper son code en 3 parties
M, V et C

XWIKI SAS

MVC :

Model

View

Controller

XEBIA NATIS

MVC :
Model = Entités

XWIKI NATIS

MVC :

Model = Entités

View = Code pour le visuel

XDNATIS

MVC :

Model = Entités

View = Code pour le visuel

Controller = Le reste

XDNATIS

MVC :

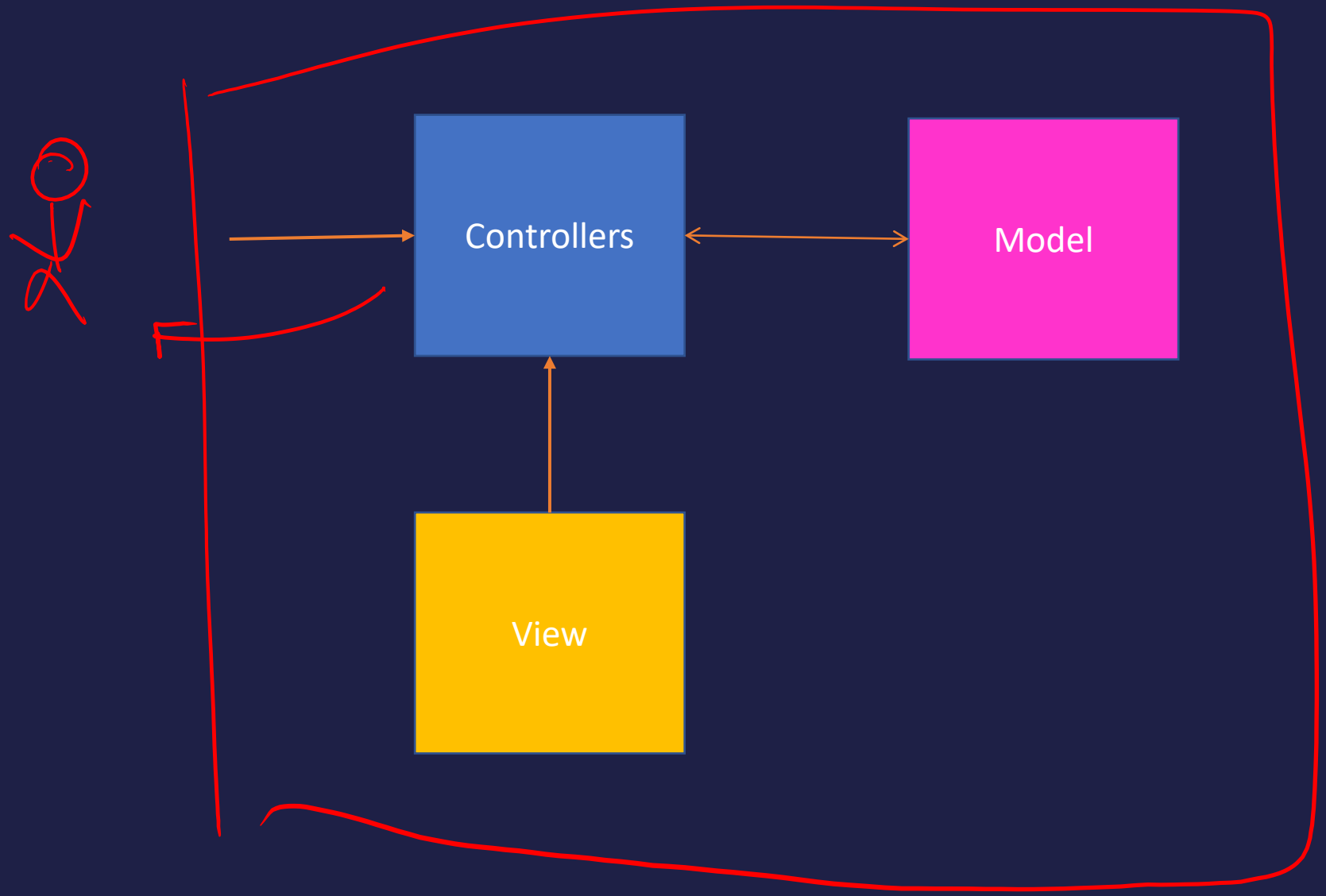
Model = Entités

View = Code pour le visuel

Controller = Le reste

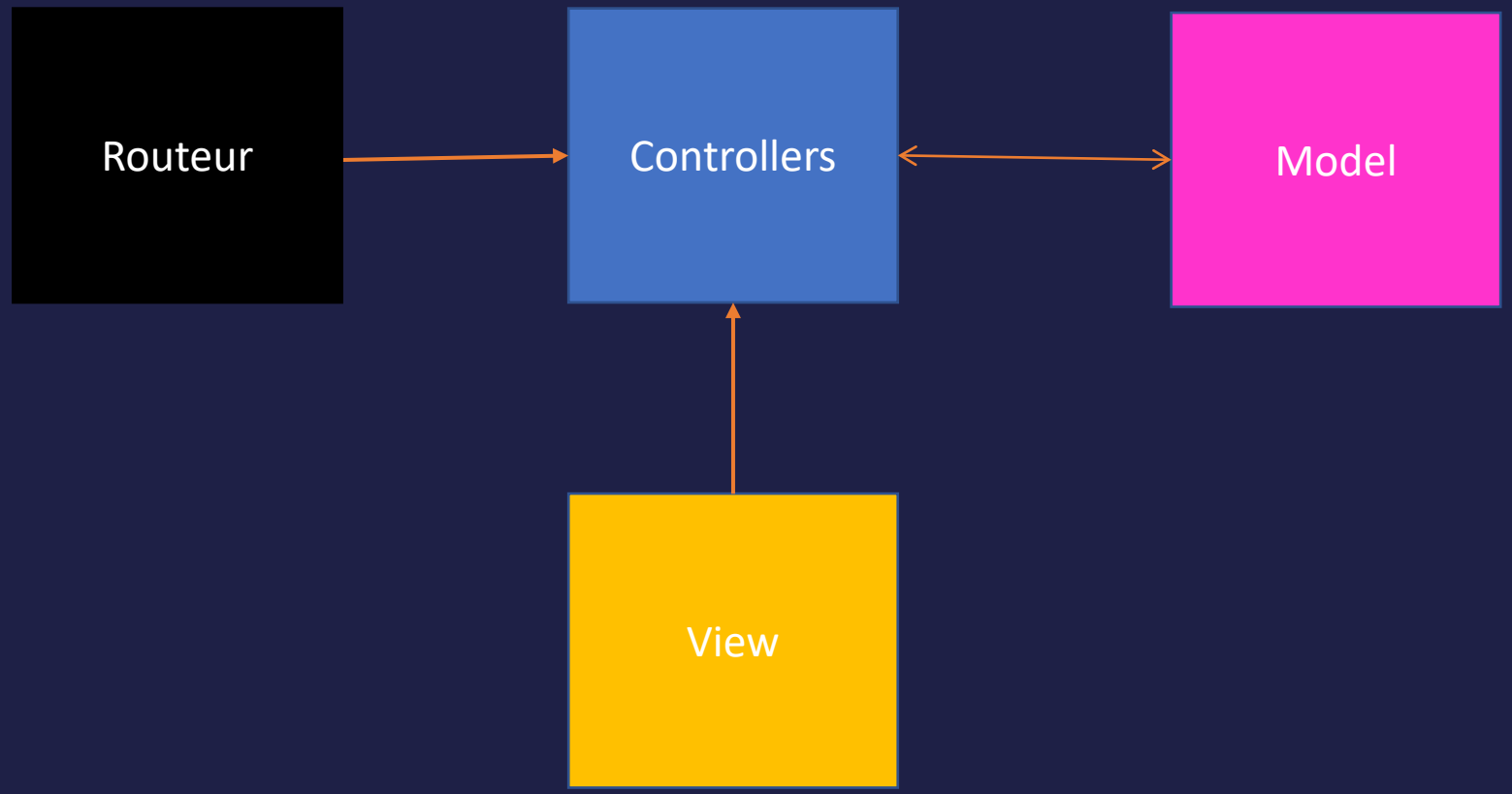
XEBIA NATIS

Architecture MVC



XDNATIS

Architecture MVC



XWIKI SAS

XWIKI SAS

XWIKI SAS

XWNATIS

UML :
DIAGRAMME DE CLASSES

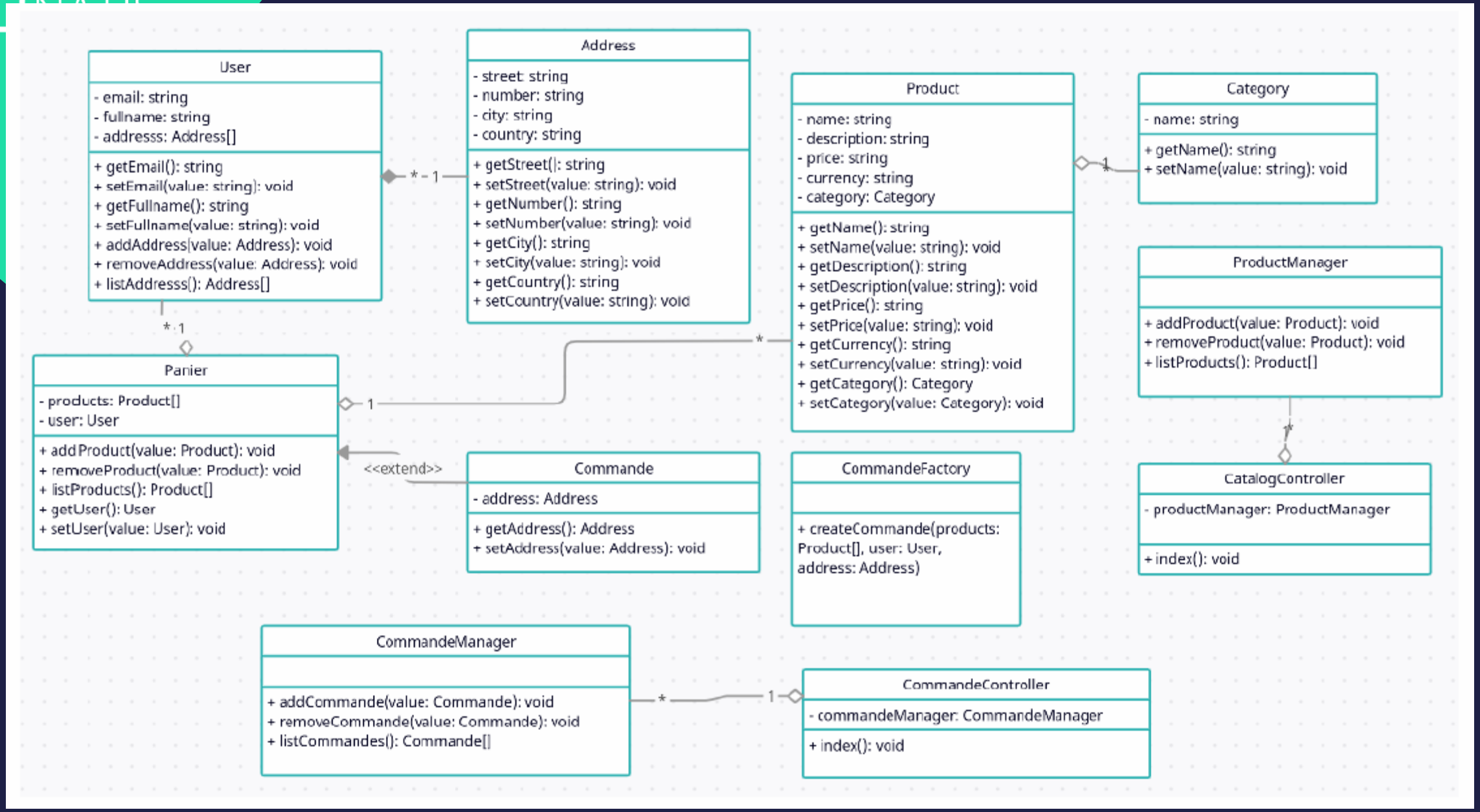
XWIKI SAS

L'UML (Unified Modeling Language), est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet.

XNATIS

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que leurs relations.

Ce diagramme fait partie de la partie statique d'UML, ne s'intéressant pas aux aspects temporels et dynamiques.


User
- email: string
- fullname: string
- addresss: Address[]
+ getEmail(): string
+ setEmail(value: string): void
+ getFullname(): string
+ setFullname(value: string): void
+ addAddress(value: Address): void
+ removeAddress(value: Address): void
+ listAddresss(): Address[]
Address
- street: string
- number: string
- city: string
- country: string
+ getStreet(): string
+ setStreet(value: string): void
+ getNumber(): string
+ setNumber(value: string): void
+ getCity(): string
+ setCity(value: string): void
+ getCountry(): string
+ setCountry(value: string): void
Product
- name: string
- description: string
- price: string
- currency: string
- category: Category
+ getName(): string
+ setName(value: string): void
+ getDescription(): string
+ setDescription(value: string): void
+ getPrice(): string
+ setPrice(value: string): void
+ getCurrency(): string
+ setCurrency(value: string): void
+ getCategory(): Category
+ setCategory(value: Category): void
Category
- name: string
+ getName(): string
+ setName(value: string): void
ProductManager
+ addProduct(value: Product): void
+ removeProduct(value: Product): void
+ listProducts(): Product[]
Panier
- products: Product[]
- user: User
+ addProduct(value: Product): void
+ removeProduct(value: Product): void
+ listProducts(): Product[]
+ getUser(): User
+ setUser(value: User): void
<<extend>>
Commande
- address: Address
+ getAddress(): Address
+ setAddress(value: Address): void
CommandeFactory
+ createCommande(products: Product[], user: User, address: Address)
CatalogController
- productManager: ProductManager
+ index(): void
CommandeManager
+ addCommande(value: Commande): void
+ removeCommande(value: Commande): void
+ listCommandes(): Commande[]
CommandeController
- commandeManager: CommandeManager
+ index(): void
* - 1
* - 1
1
*
1
*
1
*
1

XWIKI NATIS

Les classes

XEBIA NATIS

XWIKI NATIS

Les relations

XWIKI NATIS

Héritage

XWIKI NATIS

Héritage

XWIKI NATIS

Héritage
Implémentation

XWIKI NATIS

Héritage
Implémentation
Composition

XWIKI NATIS

Héritage
Implémentation
Composition
Agrégation

XDNATIS

Héritage
Implémentation
Composition
Agrégation

XWIKI NATIS

TP

XWIKI NATIS

Se mettre dans le bain
Rappel : ajout des types

XWIKI SAS

XDNATIS

Augmenter la maintenabilité avec la POO

XWIKI SAS

XWIKI NATIS

EXERCICES

XWIKI NATIS

METHODOLOGIE

1. Développer les modèles
2. Développer les providers (repositories ou managers)
3. Développer les services et/ou controllers

XWIKI NATIS

Pour les exercices suivants, nous nous limiterons à 3 entités

XWIKI NATIS

Youtube
Messenger
Doctolib
Spotify
Gmail
Booking.com
Fitbit
AppStore
Amazon
BNP Paribas

XWIKI SAS

XWIKI SAS

XWIKI SAS

XWIKI SAS

XWIKI NATIS

DESIGN PATTERNS

XWIKI NATIS

Un design pattern (patron de conception) représente une mécanique reconnue comme bonne pratique en réponse à un problème de conception.

Il décrit une solution standard et est issu de l'expérience des concepteurs.

XWIKI NATIS

https://refactoring.guru/fr/design-patterns/catalog