# Descrição dos padrões utilizados

## Adapter (composição)

- Classes:
    - Alvo: UserValidatorProtocol
    - Adaptador: UserValidatorAdapter
    - Classe existente (motivo pelo qual é necessário o adaptador): UserValidatorProtocol
- Objetivo: Facilitar a implementação de um autenticador externo, inicialmente através da API do STI. A interface UserValidatorProtocol assegura uma implementação mais segura, uma vez que a forma de autenticação poderá ser alterada futuramente e, nesse cenário, bastará apenas trabalhar com o adaptador.

## Command

- Classes:
    - Classe Cliente: CLIPrompts
    - Classe Executor: ManagersFacade
    - Classe Comando: Command
    - Classes Comando Concreto:
      ReturnBike,
      TakeBike
    - Classe Receptor: BikeManager
- Objetivo: A interface Command tem como objetivo gerar comandos mais independentes na fachada, de modo que facilite a futura adição de novos comandos.

## Facade

- Classe: ManagersFacade
- Objetivo: Estabelecer uma fachada única de comunicação com as classes de controle BikeManager, DockManager e UserManager, de modo que o usuário possa abstrair funções internas a essas classes enquanto executa ações de mais alto nível.

## Factory Method

- Classes:
    - Classes Abstrata: Creator
    - Interface: Entity
    - Classe Concreta: UserCreator, BikeCreator, DockCreator
- Objetivo: Definir uma interface comum para criar objetos, sem que o cliente precise conhecer uma classe concreta, e com isso isolar quem usa e quem cria.

## Mediator

- Classes:
    - Classe abstrata: Mediator
    - Classe concreta: UserFavoritesMediator
- Objetivo: Simplificar a integração entre o Memento e a classe de controle UserManager. Anteriormente, UserManager apresentava composição com UserFavoritesEditor e UserFavoritesCareTaker, o que piorava a legibilidade e a manutenibilidade do código. O Mediator enfraquece o acoplamento entre essas classes e estabelece a dependência a uma classe única.

## Memento

- Classes:
    - Classe Fonte: Editor
    - Classe Fonte Concreta: UserFavoritesEditor
    - Classe Memento: Memento
    - Classe Memento Concreta: UserFavoritesMemento
    - Classe Zelador: CareTaker
    - Classe Zelador Concreta: UserFavoritesCareTaker
- Objetivo: Gerar uma memória para o conjunto de bases favoritas do usuário, permitindo que um conjunto salvo anteriormente seja restaurado.

## Singleton

- Classes: SingletonBikeManager, SingletonDockManager, SingletonPBSystem, SingletonUserManager
- Objetivo: Garantir que exista apenas uma instância de cada classe de controle e de sistema, assim como fornecer um ponto de acesso global a essa instância.

## Template Method

- Classes:
    - Classe abstrata: Report
    - Classes concretas: ReportPdf, ReportTxt
- Objetivo: Definir o esqueleto de um algoritmo em uma superclasse, mas permitindo que as subclasses substituam etapas específicas do algoritmo sem modificar sua estrutura. Nesse caso, a superclasse se trata da classe “Report”, e as subclasses, de “ReportPdf” e “ReportTxt”, nas quais se consegue especificar o tipo de relatório (*report*) gerado.