Desenho detalhado - Sprint 2

Engenharia de Software Aplicada
Desenho detalhado – Sprint nº2 - Expansão Fitness

Turma : 3

Grupo nº1

nº202100299 - Rui Barroso

nº202100296 - Gonçalo Vieira

nº202100984 - Francisco Silva

nº201901953 - André Pauli

Versões **do Trabalho**

1			
2			

SUMARIO EXECUTIVO

O projeto EasyFitHub visa criar um ecossistema digital que integre eficientemente os aspectos cruciais da gestão de ginásios. Este sumário executivo destaca os três módulos principais do projeto: Autenticação, Perfil e Ginásio. Cada módulo possui requisitos específicos que serão detalhadamente abordados neste documento. O foco principal do primeiro sprint será implementar esses módulos

INTRODUÇÃO

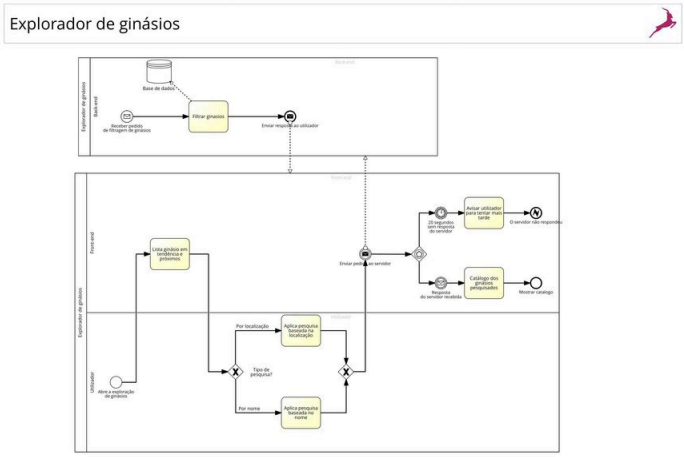
O EasyFitHub surge como resposta à necessidade crescente de gerenciamento eficiente e personalizado em ambientes de ginásio. Este documento é uma exploração detalhada dos requisitos funcionais, diagramas de classes e de estados, processos de negócio, mockups e testes de aceitação relacionados ao desenvolvimento do sistema EasyFitHub. Ao compreender e delinear esses elementos, buscamos criar uma solução tecnológica completa e eficaz para a gestão de ginásios, proporcionando aos usuários uma experiência aprimorada.

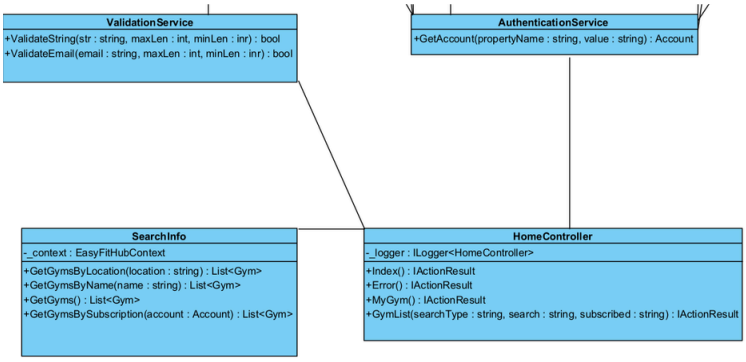
DESENHO DETALHADO

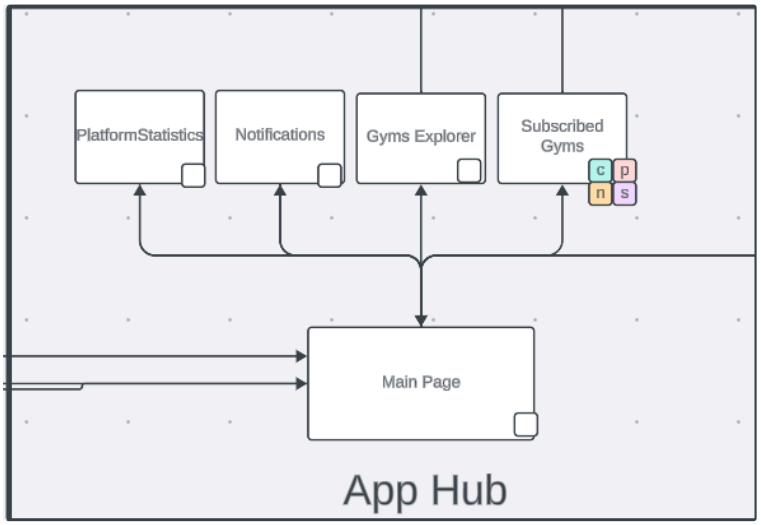
Introdução

Modulo Gym Explorer

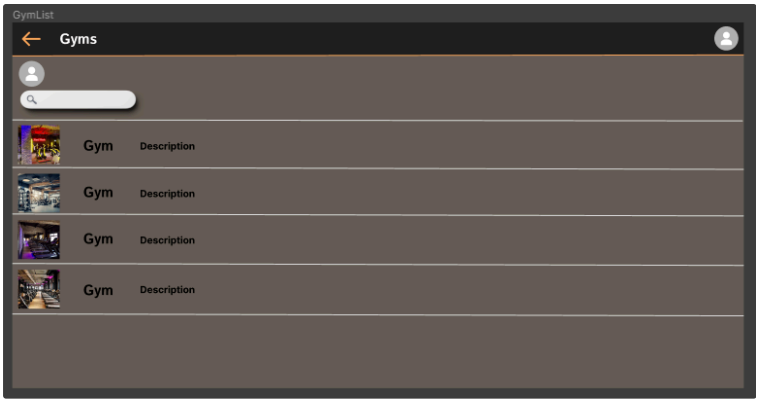
ID	Módulo	Descrição	Prioridade
R1	Explorador	O sistema deverá listar os ginásios subscritos.	Baixa
R2	Explorador	O sistema deverá permitir a pesquisa de ginásios por localização.	Baixa
R3	Explorador	O sistema deverá permitir a pesquisa de ginásios por nome.	Alta





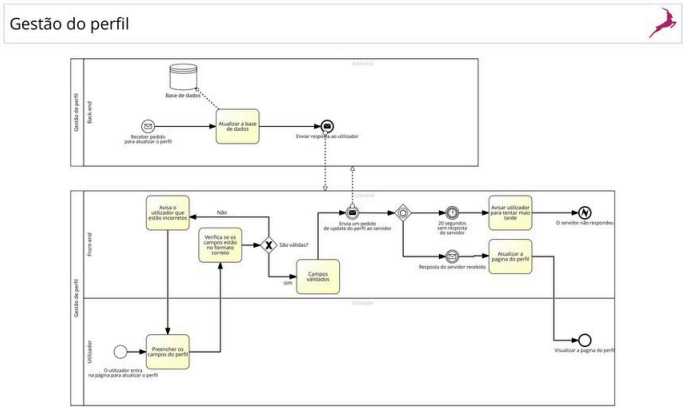


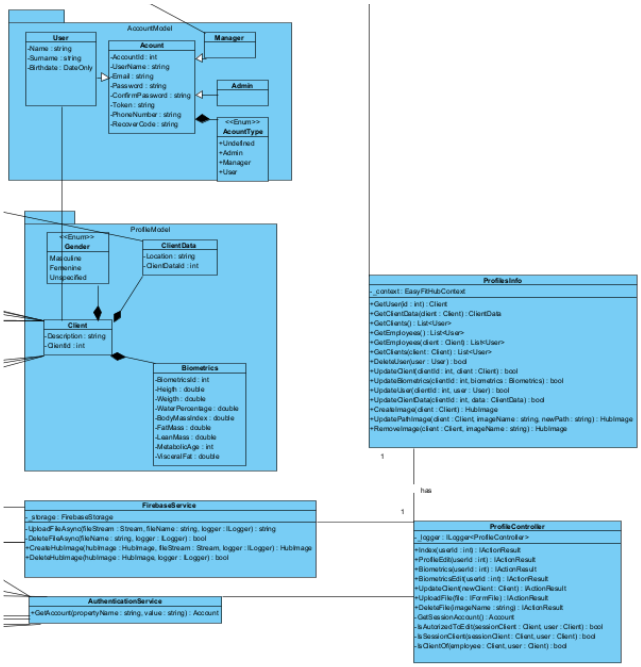
Mockup Lista ginásios

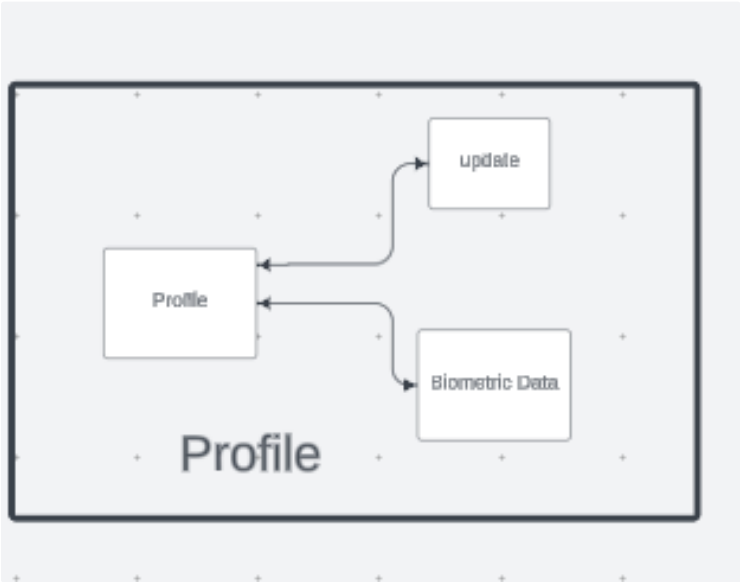


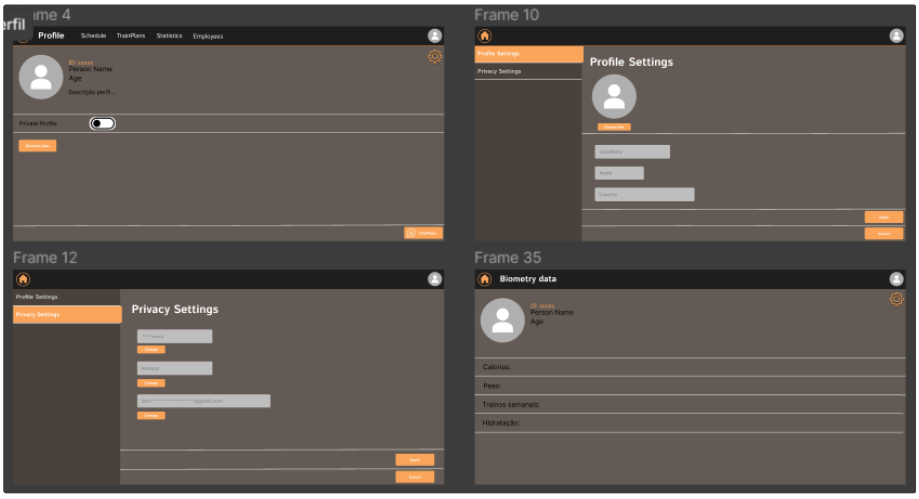
Modulo Profile

ID	Módulo	Descrição	Prioridade
R1	Perfil	O sistema deverá realizar operações CRUD no perfil.	Baixa
R2	Perfil	O sistema deverá realizar operações CRUD nos dados biométricos.	Baixa
R3	Perfil	O sistema deverá permitir os utilizadores apagarem as suas contas.	Alta





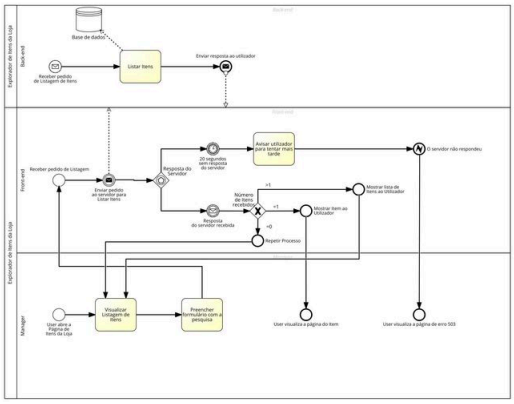




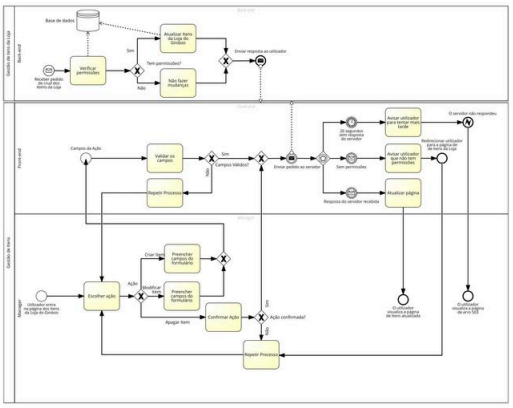
Modulo Ginásio

ID	Módulo	Descrição	Prioridade
R1	Ginásio	O sistema deverá disponibilizar uma página do ginásio.	Alta
R2	Ginásio	O sistema deverá realizar operações CRUD em itens da loja.	Baixa
R3	Ginásio	O sistema deverá realizar operações CRUD em máquinas.	Baixa
R4	Ginásio	O sistema deverá realizar operações CRUD em funcionários.	Alta
R5	Ginásio	O sistema deverá realizar operações CRUD em clientes.	Alta
R6	Ginásio	O sistema deverá permitir a gestão dos clientes de cada funcionário.	Alta
R7	Ginásio	O sistema deverá realizar operações CRUD de ginásios.	Alta

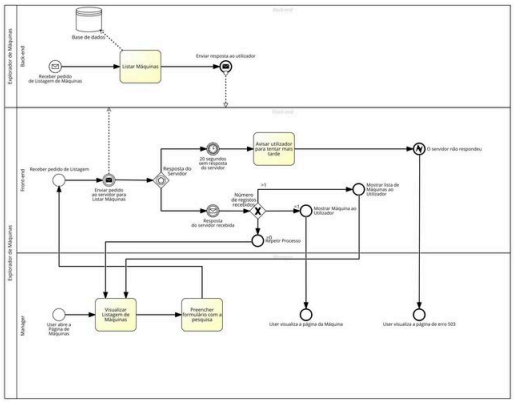
Visualizar Itens da Loja



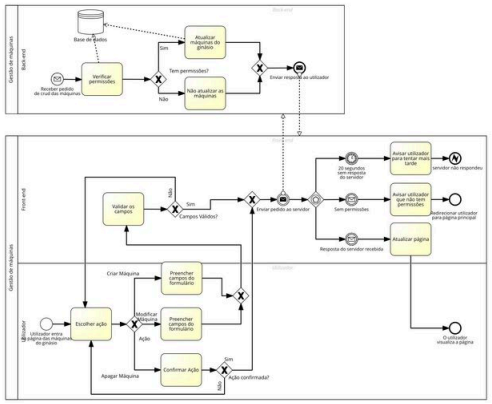
Gestão de Itens da Loja

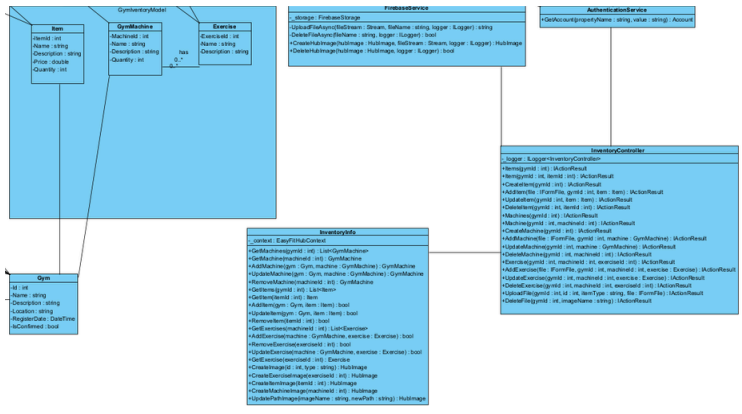


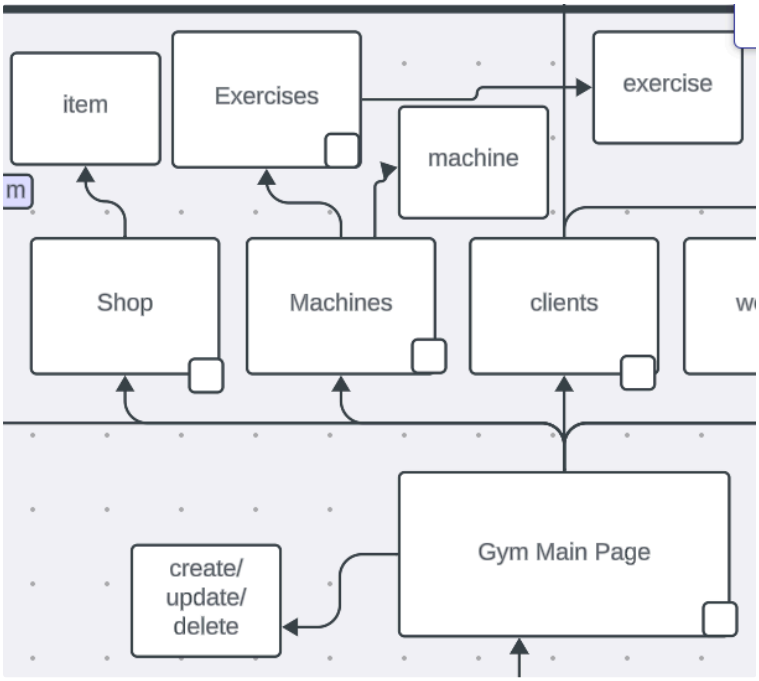
Visualizar Máquinas



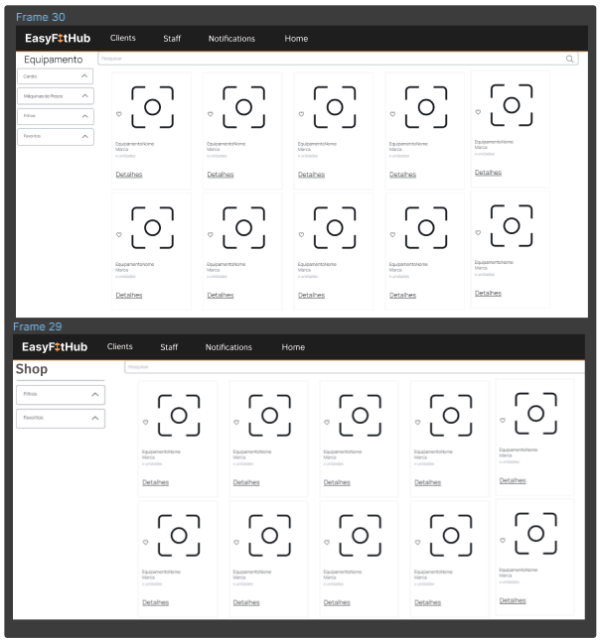
Gestão de máquinas



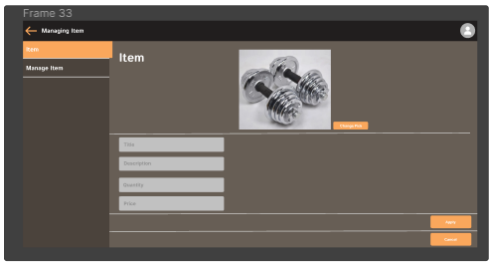




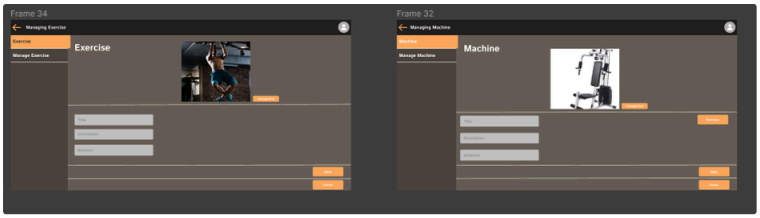
Mockup de Loja | Equipamento



Mockup Manejamento Item



Mockup Manejamento de Exercicio | Máquinas



Modulo X1/ Sprint (repetir se o Sprint tiver mais que um módulo)

Requisitos funcionais (implementados)

Testes

3.3.1 Testes Unitários

Especificação dos Casos de testes:

Nome Caso de teste:	Obter Todos os Ginásios
Código:	TC01
Finalidade:	Verificar se a função `GetGyms` do módulo "Explorador de Ginásios" retorna corretamente todos os ginásios.
Entradas:	Nenhuma
Resultados esperados:	A função deve retornar uma lista contendo todos os ginásios do sistema.
Dependências:	• A função `GetGyms` deve estar implementada. • A base de dados deve conter dados de teste representativos de vários ginásios.

Nome Caso de teste:	Pesquisar Ginásios por Localização
Código:	TC02
Finalidade:	Verificar se a função `GetGymsByLocation` do módulo "Explorador de Ginásios" retorna resultados válidos ao ser chamada com uma localização específica.
Entradas:	• Localização válida • Localização inválida
Resultados esperados:	Para a localização válida: • O resultado deve ser uma instância válida da classe Ginasio. Para a localização inválida: • A função deve informar que não há ginásios na localização especificada. • Não devem ser exibidos resultados de ginásios inexistentes.
Dependências:	• A função `GetGymsByLocation` deve estar implementado. • A base de dados de ginásios deve ser configurada corretamente para incluir dados de teste.

Nome Caso de teste:	Pesquisar Ginásios por Nome
Código:	TC03
Finalidade:	Verificar se a função `GetGymsByName` do módulo "Explorador de Ginásios" retorna resultados válidos ao ser chamada com uma localização específica.
Entradas:	• Nome de ginásio válido. • Nome de ginásio inválido.
Resultados esperados:	Para o nome válido: • O resultado deve ser uma instância válida da classe Ginasio. Para o nome inválido: • Não devem ser exibidos resultados.

Dependências:	• A função `GetGymsByName` deve estar implementada. • A base de dados deve conter dados de teste representativos de vários ginásios.

Nome Caso de teste:	Obter os Ginásios subscritos
Código:	TC04
Finalidade:	Verificar se a função `GymListBySubscription` do módulo "Explorador de Ginásios" retorna corretamente todos os ginásios a qual um utilizador esta subscrito.
Entradas:	• ID do cliente válido.
Resultados esperados:	A função deve retornar uma lista contendo todos os ginásios a que o cliente está inscrito.
Dependências:	• A função `GymListBySubscription` deve estar implementada. • A base de dados deve conter dados de teste representativos de vários ginásios e de inscrições de clientes em ginásios.

Nome Caso de teste:	Obter uma Machine a partir do seu Id
Código:	TC05
Finalidade:	Verificar se a função `GetMachine` do módulo "InventoryGym" retorna corretamente uma máquina com base no seu ID.
Entradas:	• ID de uma Maquina válido.
Resultados esperados:	A função deve retornar uma instância válida da classe Machine correspondente ao ID fornecido.
Dependências:	• A função `GetMachine` deve estar implementada no módulo "InventoryGym". • A base de dados deve conter dados de teste representativos de várias máquinas.

Nome Caso de teste:	Adicionar uma Machine nova a um Ginásio
Código:	TC06
Finalidade:	Verificar se a função `AddMachine` do módulo "InventoryGym" adiciona corretamente uma nova máquina a um ginásio do sistema.
Entradas:	• Machine a adicionar • Gym a que vai ser adicionado a maquina
Resultados esperados:	A função deve adicionar a nova máquina ao sistema e retornar - la.
Dependências:	• A função `AddMachine` deve estar implementada no módulo "InventoryGym". • A base de dados deve permitir a adição de novas máquinas.

Nome Caso de teste:	Atualizar uma Machine
Código:	TC07
Finalidade:	Verificar se a função `UpdateMachine` do módulo "InventoryGym" atualiza corretamente uma máquina.
Entradas:	• Machine nova • Gym a que vai ser alterado a maquina
Resultados esperados:	A função deve alterar máquina e retornar - la.
Dependências:	• A função `UpdateMachine` deve estar implementada no módulo "InventoryGym". • A base de dados deve conter dados de teste representativos de várias máquinas.

Nome Caso de teste:	Eliminar uma Machine a partir do seu Id
Código:	TC08
Finalidade:	Verificar se a função `RemoveMachine` do módulo "InventoryGym" elimina corretamente amáquina com base no seu ID.
Entradas:	• ID de uma Maquina válido.
Resultados	A função deve retornar a instância válida da classe Machine correspondente à eliminada.

esperados:	
Dependências:	• A função `RemoveMachine` deve estar implementada no módulo "Machine". • A base de dados deve conter dados de teste representativos de várias máquinas.

Nome Caso de teste:	Obter um Item a partir do seu Id
Código:	TC09
Finalidade:	Verificar se a função `GetItem` do módulo "InventoryGym" retorna corretamente uma item com base no seu ID.
Entradas:	• ID de um Item válido.
Resultados esperados:	A função deve retornar uma instância válida da classe Item correspondente ao ID fornecido.
Dependências:	• A função `GetItem` deve estar implementada no módulo "InventoryGym". • A base de dados deve conter dados de teste representativos de vários Itens.

Nome Caso de teste:	Adicionar um Item
Código:	TC10
Finalidade:	Verificar se a função `AddItem` do módulo "InventoryGym" adiciona corretamente um novo item ao sistema
Entradas:	• Item a adicionar • Gym a que vai ser adicionado o item
Resultados esperados:	A função deve adicionar o novo item ao ginásio e retornar o novo item.
Dependências:	• A função `AddItem` deve estar implementada no módulo "InventoryGym". • A base de dados deve permitir a adição de novos itens.

Nome Caso de teste:	Atualizar um Item
Código:	TC11
Finalidade:	Verificar se a função `UpdateItem` do módulo "InventoryGym" atualiza corretamente os dados de um item existente.
Entradas:	• Item a alterar • Gym a que vai ser adicionado o item
Resultados esperados:	A função deve alterar o item e retornar - la.
Dependências:	• A função `UpdateItem` deve estar implementada no módulo "InventoryGym". • A base de dados deve conter dados de teste representativos de diversos itens.

Nome Caso de teste:	Eliminar um Item
Código:	TC12
Finalidade:	Verificar se a função `RemoveItem` do módulo "Item" exclui corretamente um item do sistema.
Entradas:	• ID de Item válido.
Resultados esperados:	A função deve excluir o item correspondente ao ID fornecido do sistema.
Dependências:	• A função `RemoveItem` deve estar implementada no módulo "InventoryGym". • A base de dados deve conter dados de teste representativos de diversos itens.

Nome Caso de teste:	Obter um Exercise a partir do seu Id
Código:	TC13
Finalidade:	Verificar se a função `GetExercise` do módulo "InventoryGym" retorna corretamente um exercício com base no seu ID.
Entradas:	• ID de Exercise válido.
Resultados esperados:	A função deve retornar uma instância válida da classe Exercise correspondente ao ID fornecido.

Dependências:	• A função `GetExercise` deve estar implementada no módulo "InventoryGym". • A base de dados deve conter dados de teste representativos de diversos itens.

Nome Caso de teste:	Adicionar um novo Exercise
Código:	TC14
Finalidade:	Verificar se a função `AddExercise` do módulo "InventoryGym" adiciona corretamente um novo exercício ao sistema.
Entradas:	• Item a adicionar • Gym a que vai ser adicionado o item
Resultados esperados:	A função deve adicionar o novo exercício ao sistema e retornar um identificador único para o novo exercício.
Dependências:	• A função `AddExercise` deve estar implementada no módulo "InventoryGym". • O banco de dados deve permitir a adição de novos exercícios.

Nome Caso de teste:	Atualizar um Exercise
Código:	TC15
Finalidade:	Verificar se a função `UpdateExercise` do módulo "InventoryGym" atualiza corretamente os dados de um exercício existente.
Entradas:	• Exercise a alterar • Machine a que vai ser alterado o exercise
Resultados esperados:	A função deve atualizar os dados do exercício.
Dependências:	• A função `UpdateExercise` deve estar implementada no módulo "InventoryGym". • A base de dados deve conter dados de teste representativos de vários exercícios.

Nome Caso de teste:	Eliminar um Exercise
Código:	TC16
Finalidade:	Verificar se a função `RemoveExercise` do módulo "InventoryGym" exclui corretamente um exercício do sistema.
Entradas:	• ID de Exercise válido.
Resultados esperados:	A função deve remover o exercício correspondente ao ID fornecido do sistema.
Dependências:	• A função `RemoveExercise` deve estar implementada no módulo "InventoryGym". • A base de dados deve conter dados de teste representativos de vários exercícios.

Nome Caso de teste:	Atualizar ClientData
Código:	TC17
Finalidade:	Verificar se a função `UpdateClientData` do módulo "Profile" atualiza corretamente os dados de ClientData.
Entradas:	• ClientData novo • Client a ser alterado.
Resultados esperados:	A função deve atualizar os dados do ClientData correspondente a um Client.
Dependências:	• A função `UpdateClientData` deve estar implementada no módulo "Profile". • A base de dados deve conter dados de teste representativos de Client.

Nome Caso de teste:	Atualizar BiometricData
Código:	TC18
Finalidade:	Verificar se a função `UpdateBiometricData` do módulo "Profile" atualiza corretamente os dados de BiometricData.
Entradas:	• BiometricData novo • Client a ser alterado.

Resultados esperados:	A função deve atualizar os dados do BiometricData correspondente a um Client.
Dependências:	• A função `UpdateBiometricData` deve estar implementada no módulo "Profile". • A base de dados deve conter dados de teste representativos de Client.

Nome Caso de teste:	Atualizar Cliente
Código:	TC19
Finalidade:	Verificar se a função `UpdateClient` do módulo "Profile" atualiza corretamente os dados do Cliente.
Entradas:	• Client novo
Resultados esperados:	A função deve atualizar os dados do Cliente.
Dependências:	• A função `UpdateClient` deve estar implementada no módulo "Profile". • A base de dados deve conter dados de teste representativos de Clientes.

Nome Caso de teste:	Eliminar Cliente
Código:	TC20
Finalidade:	Verificar se a função `RemoveClient` do módulo "Profile" exclui corretamente um cliente do sistema.
Entradas:	• ID de Cliente válido.
Resultados esperados:	A função deve remover os dados do Cliente correspondente ao ID fornecido.
Dependências:	• A função `RemoveClient` deve estar implementada no módulo "Profile". • A base de dados deve conter dados de teste representativos de Clientes.

Especificação dos Procedimentos:

Nome Caso de teste:	Obter Todos os Ginásios
Código:	TC01
Preparação:	Configurar o ambiente de teste com dados representativos de vários ginásios na base de dados.
Inicialização:	Chamar a função `GetGyms` do módulo "Explorador de Ginásios".
Recursos específicos:	Base de dados contendo dados de teste representativos de vários ginásios.

Nome Caso de teste:	Pesquisar Ginásios por Localização
Código:	TC02
Preparação:	Configurar o ambiente de teste com dados representativos de vários ginásios na base de dados.
Inicialização:	Chamar a função `GetGymsByLocation` do módulo "Explorador de Ginásios" com uma localização válida e uma inválida..
Recursos específicos:	Base de dados contendo dados de teste representativos de vários ginásios.

Nome Caso de teste:	Pesquisar Ginásios por Nome
Código:	TC03
Preparação:	Configurar o ambiente de teste com dados representativos de vários ginásios na base de dados com diferentes nomes.
Inicialização:	Chamar a função `GetGymsByName` do módulo "Explorador de Ginásios" com um nome válido e um nome inválido.
Recursos específicos:	Base de dados contendo dados de teste representativos de vários ginásios.

Nome Caso de teste:	Obter os Ginásios subscritos
Código:	TC04
Preparação:	Configurar o ambiente de teste com dados representativos de clientes inscritos em diferentes ginásios na base de dados.
Inicialização:	Chamar a função `GymListBySubscription` do módulo "Explorador de Ginásios" com o ID de um cliente válido.
Recursos específicos:	Base de dados contendo dados de teste representativos de inscrições de clientes em ginásios.

Nome Caso de teste:	Obter uma Machine a partir do seu Id
Código:	TC05
Preparação:	Configurar o ambiente de teste com dados representativos de várias máquinas na base de dados.
Inicialização:	Chamar a função `GetMachine` do módulo "InventoryGym" com um ID de Machine válido.
Recursos específicos:	Base de dados contendo dados de teste representativos de várias máquinas.

Nome Caso de teste:	Adicionar uma Machine nova a um Ginásio
Código:	TC06
Preparação:	Configurar o ambiente de teste para permitir a adição de novas máquinas à base de dados.
Inicialização:	Chamar a função `AddMachine` do módulo "InventoryGym" com a Machine a adicionar e o Gym a que vai ser adicionado a maquina
Recursos específicos:	Base de dados configurado para permitir a adição de novas máquinas

Nome Caso de teste:	Atualizar uma Machine
Código:	TC07
Preparação:	Configurar o ambiente de teste com dados representativos de máquinas na base de dados.
Inicialização:	Chamar a função `UpdateMachine` do módulo "InventoryGym" com a nova Machine e o Gym a que vai ser atualizado a maquina
Recursos específicos:	Base de dados contendo dados de teste representativos de várias máquinas.

Nome Caso de teste:	Eliminar uma Machine a partir do seu Id
Código:	TC08
Preparação:	Configurar o ambiente de teste com dados representativos de máquinas na base de dados.
Inicialização:	Chamar a função `RemoveMachine` do módulo "InventoryGym" com ID de uma Maquina válido.
Recursos específicos:	Base de dados contendo dados de teste representativos de várias máquinas.

Nome Caso de teste:	Obter um Item a partir do seu Id
Código:	TC09
Preparação:	Configurar o ambiente de teste com dados representativos de diversos itens no banco de dados.
Inicialização:	Chamar a função `GetItem` do módulo "InventoryGym" com o ID de Item válido.
Recursos específicos:	Base de dados contendo dados de teste representativos de diversos itens.

Nome Caso de teste:	Adicionar um Item
Código:	TC10

Preparação:	Configurar o ambiente de teste para permitir a adição de novos itens ao banco de dados.
Inicialização:	Chamar a função `AddItem` do módulo "InventoryGym" com o Item a adicionar e o Gym a que vai ser adicionado o item
Recursos específicos:	Base de dados configurado para permitir a adição de novos itens.

Nome Caso de teste:	Atualizar um Item
Código:	TC11
Preparação:	Configurar o ambiente de teste com dados representativos de diversos itens no banco de dados.
Inicialização:	Chamar a função `UpdateItem` do módulo "InventoryGym" com o Item a alterar e o Gym a que vai ser alterado o item
Recursos específicos:	Base de dados contendo dados de teste representativos de diversos itens.

Nome Caso de teste:	Eliminar um Item
Código:	TC12
Preparação:	Configurar o ambiente de teste com dados representativos de diversos itens no banco de dados.
Inicialização:	Chamar a função `RemoveItem` do módulo "InventoryGym" com o ID de Item válido.
Recursos específicos:	Base de dados contendo dados de teste representativos de diversos itens.

Nome Caso de teste:	Obter um Exercise a partir do seu Id
Código:	TC13
Preparação:	Configurar o ambiente de teste com dados representativos de vários exercícios na base de dados.
Inicialização:	Chamar a função `GetExercise` do módulo "InventoryGym" com o ID de Exercise válido.
Recursos específicos:	Base de dados contendo dados de teste representativos de vários exercícios.

Nome Caso de teste:	Adicionar um novo Exercise
Código:	TC14
Preparação:	Configurar o ambiente de teste com dados representativos de vários exercícios na base de dados.
Inicialização:	Chamar a função `AddExercise` do módulo "InventoryGym" com o Item a adicionar e o Gym a que vai ser adicionado o item
Recursos específicos:	Base de dados configurado para permitir a adição de novos exercícios.

Nome Caso de teste:	Atualizar um Exercise
Código:	TC15
Preparação:	Configurar o ambiente de teste com dados representativos de vários exercícios na base de dados.
Inicialização:	Chamar a função `UpdateExercise` do módulo "InventoryGym" com o novo Exercise e a Machine a que vai ser alterado o exercise
Recursos específicos:	Base de dados contendo dados de teste representativos de vários exercícios.

Nome Caso de teste:	Eliminar um Exercise
Código:	TC16
Preparação:	Configurar o ambiente de teste com dados representativos de vários exercícios na base de dados.
Inicialização:	Chamar a função `RemoveExercise` do módulo "InventoryGym" com o ID de Exercise válido.
Recursos específicos:	Base de dados contendo dados de teste representativos de vários exercícios.

Nome Caso de teste:	Atualizar ClientData
Código:	TC17
Preparação:	Configurar o ambiente de teste com dados representativos de Clientes na base de dados.
Inicialização:	Chamar a função `UpdateClientData` do módulo "Explorador de Ginásios" com ClientData novo e o Client a ser alterado.
Recursos específicos:	Base de dados contendo dados de teste representativos de Clientes.

Nome Caso de teste:	Atualizar BiometricData
Código:	TC18
Preparação:	Configurar o ambiente de teste com dados representativos de Clientes na base de dados.
Inicialização:	Chamar a função `UpdateBiometricData` do módulo "Explorador de Ginásios" com BiometricDatanovo e o Client a ser alterado.
Recursos específicos:	Base de dados contendo dados de teste representativos de Clientes.

Nome Caso de teste:	Atualizar Cliente
Código:	TC19
Preparação:	Configurar o ambiente de teste com dados representativos de Clientes na base de dados.
Inicialização:	Chamar a função `UpdateClient` do módulo "Explorador de Ginásios" Client novo.
Recursos específicos:	Base de dados contendo dados de teste representativos de Clientes.

Nome Caso de teste:	Eliminar Cliente
Código:	TC20
Preparação:	Configurar o ambiente de teste com dados representativos de Clientes na base de dados.
Inicialização:	Chamar a função `RemoveClient` do módulo "Profile" com ID de Cliente válido.
Recursos específicos:	Base de dados contendo dados de teste representativos de Clientes.

Testes de Automação

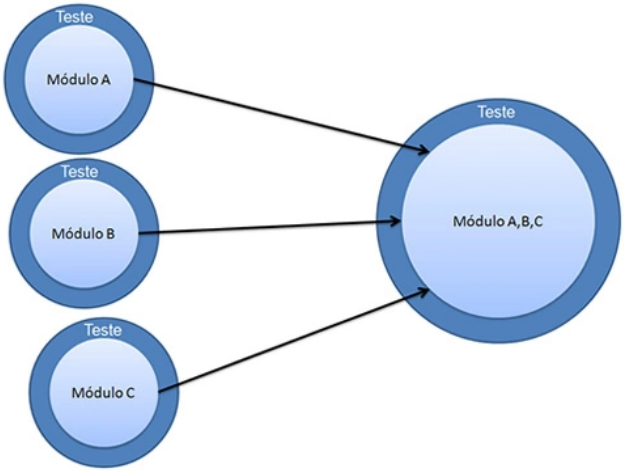
(Validar a navegação e formulários com selenium ou katalon)

Testes de Integração

O teste de integração que iremos utilizar será o BigBang.

O teste de integração de big bang é uma abordagem de teste de software em que todos os componentes ou módulos do sistema são integrados simultaneamente para testar a sua funcionalidade como um todo.

Ao realizar o teste, todos os módulos serão combinados e testados em conjunto para verificar se o sistema como um todo funciona corretamente.


Teste
Módulo A
Teste
Módulo B
Teste
Módulo C
Teste
Módulo A,B,C