CURSO SUPERIOR DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
UNIVERSIDADE POSITIVO - LONDRINA 2021

Alunos: Bruno Fukunaga, Iris Sanches Borges, Leonardo Mandelli Gonçalves, Nicolas Silva Cavalcante Souza

Projeto Integrador

Desenvolvimento de Aplicativos Móveis, Desenvolvimento de Software Visual, Arquitetura de Software
Professoras: Liliane Kashiwabara e Juliana Costa Silva

ATIVIDADE 01 - DEFINIÇÃO DO PROJETO

- Gerenciamento de Consultas Médicas (Mobile)

- Resumo:
  O Projeto consiste em desenvolver uma aplicação mobile de gerenciamento de consultas médicas para uma clínica médica, a aplicação deve permitir o cadastro e gerenciamento de: pacientes, especialistas e o agendamento e registro das consultas.

- Épicos:
  - Gestão de pacientes;
  - Gestão de especialistas;
  - Gestão de consultas;

ATIVIDADE 2 - LEVANTAMENTO DE ENTIDADES, REQUISITOS E DER

1. Levantamento de entidades

Entidade	Atributos
Paciente	- Nome Paciente - CPF - Endereco - Telefone - Data Nascimento - Plano Saúde (true / false)
Especialista	- Nome Especialista - CRM - Especialização - Exige Plano Saúde (true / false) - Telefone - Email
Consulta	- Data - Horário - Nome Paciente - Nome Especialista - Código Consulta

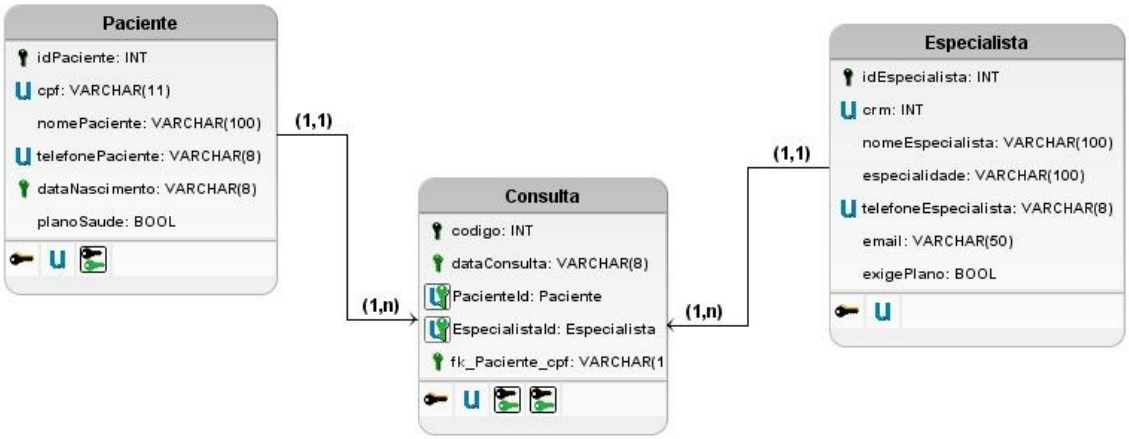
2. Levantamento de requisitos

Número do Requisito	Descrição do Requisito	Regras
1	O sistema deverá permitir a inclusão e alteração de pacientes	- O campo nome não pode ser vazio - O campo nome deve conter no mínimo 5 e no máximo 100 caracteres - O CPF é obrigatório com a sua validação - Não pode conter dois ou mais CPF iguais - O campo plano de saúde é obrigatório com apenas SIM ou NÃO (true / false) com a sua validação.

		- Data de Nascimento deve ser anterior ao dia do cadastro
2	O sistema deverá permitir a busca do paciente pelo seu CPF	- Caso não exista no banco de dados o CPF informado, o sistema deverá apresentar a seguinte mensagem: “CPF digitado não existe!”
3	O sistema deverá permitir a inclusão e alteração de especialistas	- O campo nome não pode ser vazio - O campo nome deve conter no mínimo 5 e no máximo 100 caracteres - O CRM é obrigatório com a sua validação - Não pode conter dois ou mais CRM iguais - O campo especialização é obrigatório - Cada especialista pode desempenhar apenas uma especialização - O campo exige Plano é obrigatório com apenas SIM ou NÃO (true / false) com a sua validação.
4	O sistema deverá permitir a busca do especialista pelo seu CRM	- Caso não exista no banco de dados o CRM informado, o sistema deverá apresentar a seguinte mensagem: “CRM digitado não existe!”
5	O sistema deverá permitir a inclusão e alteração de consultas	- O campo código é obrigatório - Não pode haver dois ou mais códigos com os mesmos valores - O campo nome especialista é obrigatório com a sua validação - O campo nome paciente é obrigatório com a sua validação - O campo data é obrigatório - O campo horário é obrigatório
6	O sistema deverá permitir a busca da consulta pelo seu código	- Caso não exista no banco de dados o código informado, o sistema deverá apresentar a seguinte mensagem: “Código digitado não existe!”
7	O sistema deverá permitir a exclusão de consultas (Desmarcar Consulta)	- A exclusão será feita através do código da consulta - Caso o código digitado não se referir a nenhum cadastro, o sistema deve apresentar a seguinte mensagem:

		“Não foi possível realizar a exclusão da consulta, pois o código informado não foi cadastrado”

3. DER – Diagrama de entidades relacionamento



ATIVIDADE 3 - DEFINIÇÃO DE TAREFAS

Número do Requisito	Descrição do Requisito	Tarefas
N/A	Para atender todos os requisitos do sistema, será necessário	- Criar a ambientação da linguagem C# - Criação dos Profiles - Criar repositório no Git - Instalação dos pacotes necessários - Configurar a conexão com o banco de dados MySQL

		- Scripts para a criação das tarefas - Mapear as entidades
1	O sistema deverá permitir a inclusão e alteração de pacientes	- Salvar os dados do paciente (endpoint C#) - Criação da tela de entrada de dados do paciente (Cadastrar Paciente) - Desenvolver os endpoints da aplicação
2	O sistema deverá permitir a busca do paciente pelo seu CPF	- Criar a tela que permita que o usuário digite o CPF do paciente desejado - Desenvolver os endpoints da aplicação
3	O sistema deverá permitir a inclusão e alteração de especialistas	- Salvar os dados do especialista (endpoint C#) - Criação da tela de entrada de dados do especialista (Cadastrar Especialista) - Desenvolver os endpoints da aplicação
4	O sistema deverá permitir a busca do especialista pelo seu CRM	- Criar a tela que permita que o usuário digite o CRM do especialista desejado - Desenvolver os endpoints da aplicação
5	O sistema deverá permitir a inclusão e alteração de consultas	- Salvar os dados da consulta (endpoint C#) - Criação da tela de entrada de dados da consulta

		(Cadastrar Consulta) - Desenvolver os endpoints da aplicação
6	O sistema deverá permitir a busca da consulta pelo seu código	- Criar a tela que permita que o usuário digite o código da consulta desejada - Desenvolver os endpoints da aplicação
7	O sistema deverá permitir a exclusão de consultas (Desmarcar Consulta)	- Criar a tela que permita que o usuário digite o código da consulta deseja excluir - Desenvolver os endpoints da aplicação

ATIVIDADE 4 - PROTÓTIPOS DE TELA

1. Telas

Figura 1: Tela Inicial.

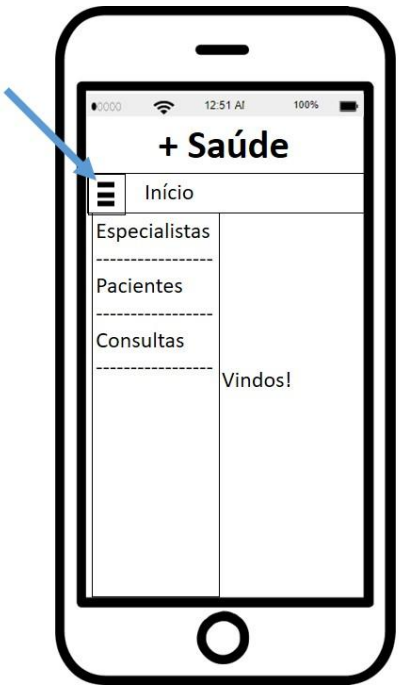


Figura 2: Menu de navegação.

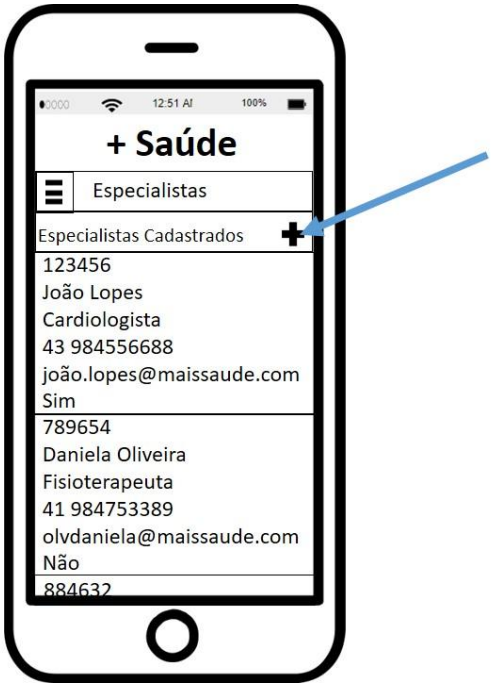


Figura 3: Tela especialistas, mostra a lista de especialistas cadastrados e mais opções.

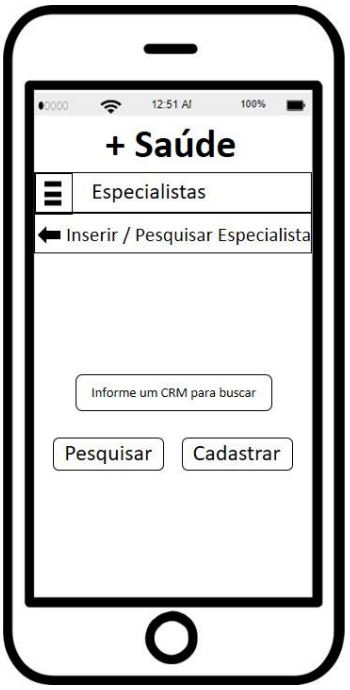


Figura 4: Mais opções, dá a opção de buscar um especialista pelo seu CRM ou cadastrar um novo especialista.

Figura 5: Tela para cadastro do novo especialista

Figura 6: Tela para editar especialista, ao clicar sob um especialista da lista podemos editar os seus dados ou excluí-lo.

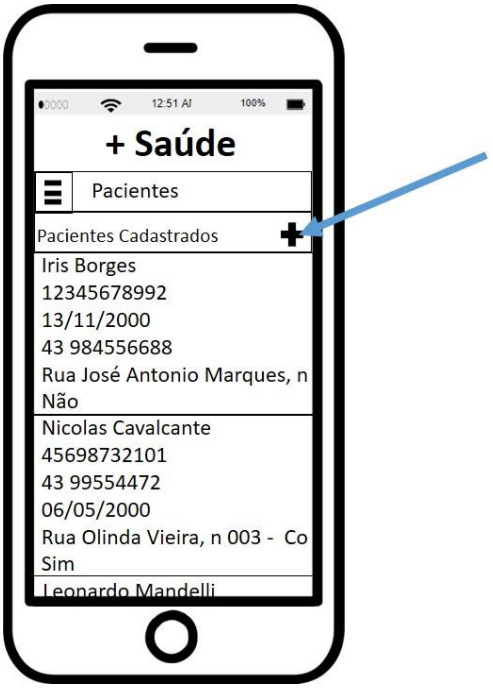


Figura 7: Tela pacientes, mostra a lista de pacientes cadastrados e mais opções.

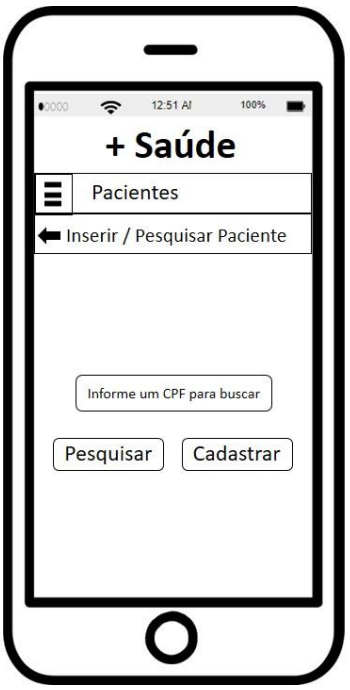


Figura 8: Mais opções, dá a opção de buscar um paciente pelo seu CPF ou cadastrar um novo paciente.

Figura 9: Tela para cadastro do novo paciente.

Figura 10: Tela para editar paciente, ao clicar sob um paciente da lista podemos editar os seus dados ou excluí-lo.

Figura 11: Tela Consultas, mostra a lista de consultas agendadas e mais opções.

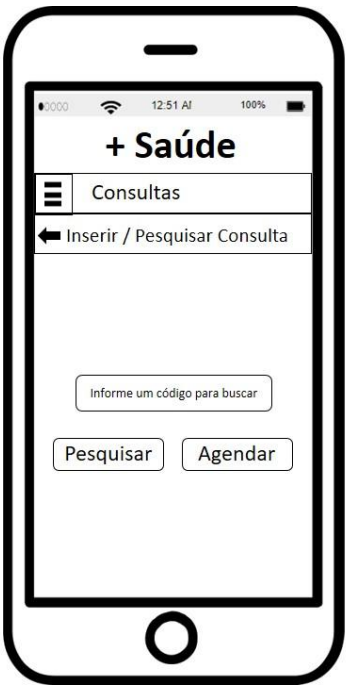


Figura 12: Mais opções, dá a opção de buscar uma consulta pelo seu código ou agendar uma nova consulta.

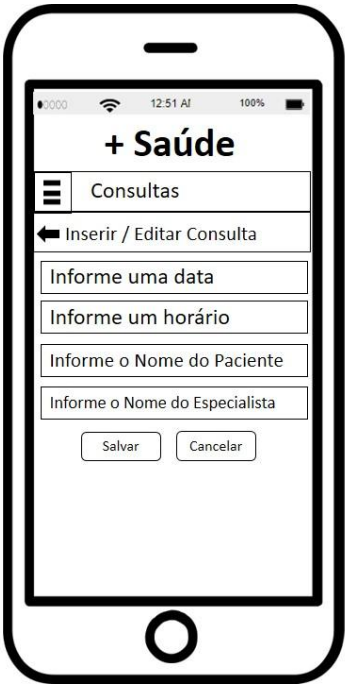
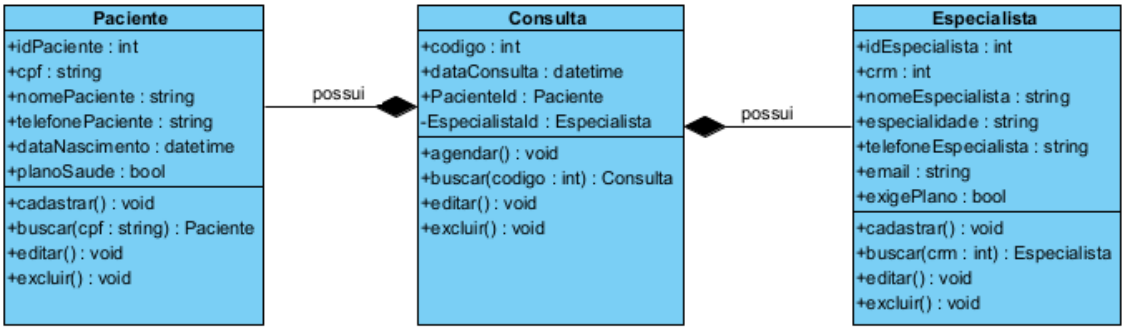


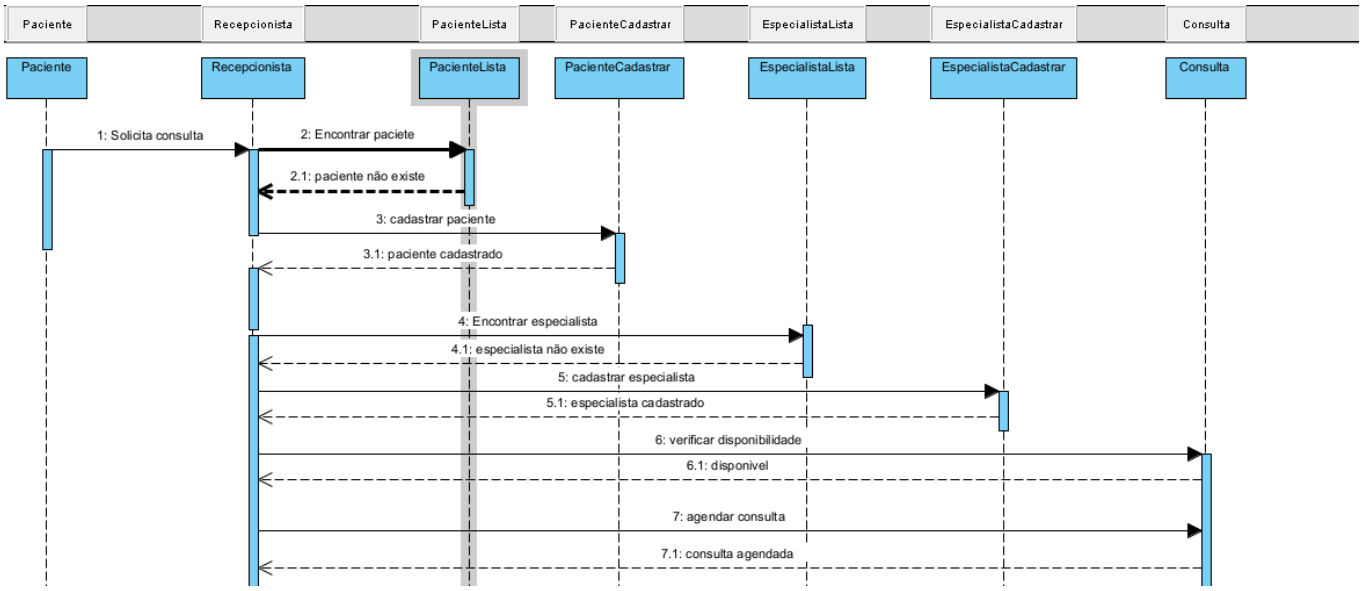
Figura 13: Tela para agendar nova consulta.

Figura 14: Tela para editar consulta, ao clicar sob uma consulta da lista podemos editar os seus dados ou desmarcá-la.

2. Diagrama de classes do sistema



3. Lista diagramas



Unidade de teste	Casos de testes
PacienteService	Verificar o comportamento do método pacienteService.buscaPorCpf quando o retorno do pacienteRepository.findByCpf for um objeto paciente carregado.
	Verificar o comportamento do método pacienteService.buscaPorCpf quando o retorno do pacienteRepository.findByCpf for vazio.
	Verificar o comportamento do método pacienteService.salvar quando o método pacienteRepository.save for executado com sucesso.
	Verificar o comportamento do método pacienteService.salvar quando o retorno do pacienteRepository.findByCpf for vazio.
	Verificar o comportamento do método pacienteService.salvar quando o retorno do pacienteRepository.save for uma exceção do tipo DataIntegrityViolationException, que indica que uma regra de chave única foi violada (no caso, o cpf que está sendo cadastrado já existe no banco).

PacienteController	Verificar o comportamento do método PacienteController.buscaPorCpf quando o retorno do pacienteService.buscaPorCpf for um objeto paciente carregado.
	Verificar o comportamento do método PacienteController.buscaPorCpf quando o retorno do

	pacienteService.buscaPorCpf for uma exceção do tipo ConsistenciaException.
	Verificar o comportamento do método PacienteController.salvar quando o método pacienteService.salvar for executado com sucesso.
	Verificar o comportamento do método PacienteController.salvar quando o método pacienteService.salvar for uma exceção do tipo ConsistenciaException.
	Verificar o comportamento do método pacienteService.salvar quando o nome do paciente não for informado.
	Verificar o comportamento do método pacienteService.salvar quando o nome do paciente possuir menos que 5 caracteres.
	Verificar o comportamento do método pacienteService.salvar quando o nome do paciente possuir mais que 100 caracteres.
	Verificar o comportamento do método pacienteService.salvar quando o cpf do paciente não for informado.
	Verificar o comportamento do método pacienteService.salvar quando o cpf do paciente não for válido.
	Verificar o comportamento do método pacienteService.salvar quando o plano de saúde do paciente não for informado (true/false).
	Verificar o comportamento do método pacienteService.salvar quando o endereço do paciente não for informado.
	Verificar o comportamento do método pacienteService.salvar quando o endereço do paciente não for válido.
	Verificar o comportamento do método pacienteService.salvar quando a data de nascimento do paciente não for informada.
	Verificar o comportamento do método pacienteService.salvar quando a data de nascimento do paciente não for válida (não for anterior à data de cadastro).
	Verificar o comportamento do método pacienteService.salvar quando o telefone do paciente não for informado.
	Verificar o comportamento do método pacienteService.salvar quando o telefone do paciente for diferente de 11 dígitos.

EspecialistaService	Verificar o comportamento do método especialistaService.buscaPorCrm quando o retorno do especialistaRepository.findByCrm for um objeto especialista carregado.
	Verificar o comportamento do método especialistaService.buscaPorCrm quando o retorno do especialistaRepository.findByCrm for vazio.
	Verificar o comportamento do método especialistaService.salvar quando o método especialistaRepository.save for executado com sucesso.
	Verificar o comportamento do método especialistaService.salvar quando o retorno do especialistaRepository.findByCrm for vazio.
	Verificar o comportamento do método especialistaService.salvar quando o retorno do especialistaRepository.save for uma exceção do tipo DataIntegrityViolationException, que indica que uma regra de chave única foi violada (no caso, o crm que está sendo cadastrado já existe no banco).

EspecialistaController	Verificar o comportamento do método EspecialistaController.buscaPorCrm quando o retorno do especialistaService.buscaPorCrm for um objeto especialista carregado.
	Verificar o comportamento do método EspecialistaController.buscaPorCrm quando o retorno do especialistaService.buscaPorCrm for uma exceção do tipo ConsistenciaException.
	Verificar o comportamento do método EspecialistaController.salvar quando o método especialistaService.salvar for executado com sucesso.
	Verificar o comportamento do método EspecialistaController.salvar quando o método especialistaService.salvar for uma exceção do tipo ConsistenciaException.
	Verificar o comportamento do método especialistaService.salvar quando o nome do especialista não for informado.
	Verificar o comportamento do método especialistaService.salvar quando o nome do especialista

	possuir menos que 5 caracteres.
	Verificar o comportamento do método especialistaService.salvar quando o nome do especialista possuir mais que 100 caracteres.
	Verificar o comportamento do método especialistaService.salvar quando o crm do especialista não for informado.
	Verificar o comportamento do método especialistaService.salvar quando o crm do especialista for menor que 4 dígitos.
	Verificar o comportamento do método especialistaService.salvar quando o crm do especialista for maior que 6 dígitos.
	Verificar o comportamento do método especialistaService.salvar quando a especialização do especialista não for informada.
	Verificar o comportamento do método especialistaService.salvar quando a especialização do especialista não for válida (mais de 1).
	Verificar o comportamento do método especialistaService.salvar quando o telefone do especialista não for informado.
	Verificar o comportamento do método especialistaService.salvar quando o telefone do especialista for diferente de 11 dígitos.
	Verificar o comportamento do método especialistaService.salvar quando o email do especialista não for informado.
	Verificar o comportamento do método especialistaService.salvar quando o email do especialista não for válido (quando não for no formato correto de um email).

ConsultaService	Verificar o comportamento do método consultaService.buscaPorCodigoConsulta quando o retorno do consultaRepository.findById for um objeto consulta carregado.
	Verificar o comportamento do método consultaService.buscaPorCodigoConsulta quando o

	retorno do consultaRepository.findById for vazio.
	Verificar o comportamento do método consultaService.salvar quando o método consultaRepository.save for executado com sucesso.
	Verificar o comportamento do método consultaService.salvar quando o retorno do consultaRepository.findById for vazio.
	Verificar o comportamento do método consultaService.salvar quando o retorno do consultaRepository.save for uma exceção do tipo DataIntegrityViolationException, que indica que uma regra de chave única foi violada (no caso, o código da consulta que está sendo cadastrado já existe no banco).

ConsultaController	Verificar o comportamento do método ConsultaController.buscaPorCodigoConsulta quando o retorno do consultaService.buscaPorId for um objeto consulta carregado.
	Verificar o comportamento do método ConsultaController.buscaPorCodigoConsulta quando o retorno do consultaService.buscaPorId for uma exceção do tipo ConsistenciaException.
	Verificar o comportamento do método ConsultaController.deletar quando o método consultaService.deletar for executado com sucesso.
	Verificar o comportamento do método ConsultaController.deletar quando o retorno do consultaService.deletar for uma exceção do tipo ConsistenciaException.
	Verificar o comportamento do método ConsultaController.salvar quando o método consultaService.salvar for uma exceção do tipo ConsistenciaException.
	Verificar o comportamento do método consultaService.salvar quando a data da consulta não for informada.
	Verificar o comportamento do método consultaService.salvar quando a data da consulta não for válida (anterior à data de cadastro).
	Verificar o comportamento do método

	consultaService.salvar quando o código da consulta não for informado.
	Verificar o comportamento do método consultaService.salvar quando o nome do especialista não for informado.
	Verificar o comportamento do método consultaService.salvar quando o nome do especialista possuir menos que 5 caracteres.
	Verificar o comportamento do método consultaService.salvar quando o nome do especialista possuir mais que 100 caracteres.
	Verificar o comportamento do método consultaService.salvar quando o nome do paciente não for informado.
	Verificar o comportamento do método consultaService.salvar quando o nome do paciente possuir menos que 5 caracteres.
	Verificar o comportamento do método consultaService.salvar quando o nome do paciente possuir mais que 100 caracteres.
	Verificar o comportamento do método consultaService.salvar quando o horário da consulta não for informado.
	Verificar o comportamento do método consultaService.salvar quando o horário da consulta não for válido (diferente do formato hh:mm).