Documentação API - SysOdonto

Equipe:
Cauã Grigolatto Domingos – AQ3022323 – caua.grigolatto@aluno.ifsp.edu.br - Líder
Gabriel de Pauli Santos – AQ302282X – gabriel.pauli@aluno.ifsp.edu.br Gabriel Dellatore Ezequiel - AQ3022561 - dellatore.gabriel@aluno.ifsp.edu.br Gabriel Ventura Pires – AQ3023672 – g.ventura@aluno.ifsp.edu.br João Pedro da Silva Vieira - AQ3022366 - vieira.joao1@aluno.ifsp.edu.br

Repositório GitHub1 : https://github.com/projetologicas/PRSI_2025_SysOdonto

1. POST /view/auth/login - Login de Usuário

Descrição: Realiza a autenticação do usuário com e-mail e senha. Se as credenciais estiverem corretas, gera um token JWT e o adiciona como um cookie de nome `jwt` na resposta. Após o login, retorna o objeto do usuário logado e inicia o serviço de bot do WhatsApp.

Método HTTP: `POST`

Endpoint: `/view/auth/login`

Corpo da Requisição (JSON) - `AuthRequest`:

```
{
  "email": "dentista@exemplo.com",
  "password": "suaSenhaSecreta"
}
```

Códigos de Resposta:

- `200 OK`: Login realizado com sucesso. Retorna o JSON contendo `message` e o objeto `loggedUser`.
- `401 Unauthorized`: Credenciais inválidas (E-mail ou senha incorretos). Retorna um JSON com `error`.
- `500 Internal Server Error`: Erro inesperado no servidor.

2. POST /view/auth/register - Cadastro de Usuário

Descrição: Cadastra um novo usuário (dentista) no sistema. Verifica se as senhas coincidem e se o e-mail já está em uso. Se o cadastro for bem-sucedido, gera o token JWT e o adiciona como cookie `jwt` na resposta, retornando os dados do novo usuário.

Método HTTP: `POST`

Endpoint: `/view/auth/register`

Corpo da Requisição (JSON) - `RegisterRequest`:

```
{
  "name": "Nome do Dentista",
  "email": "novo@exemplo.com",
  "password": "senha",
  "confirmPassword": "senha"
}
```

Códigos de Resposta:

- `200 OK`: Cadastro realizado com sucesso. Retorna o JSON contendo `message` e o objeto `loggedUser`.
- `400 Bad Request`: As senhas fornecidas em `password` e `confirmPassword` não coincidem.
- `409 Conflict`: O e-mail fornecido já está cadastrado no sistema (`EmailAlreadyUsedException`).
- `500 Internal Server Error`: Erro ao persistir os dados do cadastro ou erro inesperado.

3. GET /view/auth/{email} - Obter Usuário Logado

Descrição: Retorna os dados completos do usuário associado ao e-mail fornecido na URL. Este endpoint é usado internamente, como após um login ou para checar a sessão.

Método HTTP: `GET`

Endpoint: `/view/auth/{email}`

Exemplo de Resposta (JSON) - `User`:

```
{
  "id": "UmIDGerado",
  "email": "dentista@exemplo.com",
  "name": "Nome do Usuário",
  "profilePicture": "url_da_foto",
  "password": "senha_hash"
}
```

Códigos de Resposta:

- `200 OK`: Usuário encontrado. Retorna o objeto `User`.
- `404 Not Found`: Nenhum usuário encontrado com o e-mail especificado.
- `500 Internal Server Error`: Erro ao acessar a camada de serviço/dados.

4. POST /view/patients - Cadastro de Paciente

Descrição: Cadastra um novo paciente no banco de dados. O paciente é atrelado ao usuário que está logado via `userId`. Campos como `name`, `cpf` (formato `xxx.xxx.xxx-xx`) e `telephone` são obrigatórios.

Método HTTP: `POST`

Endpoint: `/view/patients`

Corpo da Requisição (JSON) - `Patient`:

```
{
  "name": "Maria Silva",
  "cpf": "123.456.789-00",
  "telephone": "11999998888",
  "birthDate": "1985-10-25",
  "startTreatmentDate": "2025-01-10",
  "observations": "Paciente com histórico de sensibilidade."
}
```

Códigos de Resposta:

- `200 OK`: Paciente cadastrado com sucesso.

- `400 Bad Request`: Falha na validação dos dados (e.g., CPF ou telefone com formato inválido, campos obrigatórios vazios, data de nascimento futura).

5. POST /view/consultations - Cadastrar Nova Consulta

Descrição: Agenda uma nova consulta para um paciente existente. O campo `dateTime` deve ser uma data e hora futura ou presente. O sistema pode verificar conflitos de horário.

Método HTTP: `POST`

Endpoint: `/view/consultations`

Corpo da Requisição (JSON) - `Consultation`:

```
{
  "patientId": "IDDoPaciente",
  "dateTime": "2025-12-30T10:00",
  "patientName": "Nome do Paciente",
  "observations": "Retorno para revisão de cárie.",
  "sendReminder": true
}
```

Códigos de Resposta:

- `200 OK`: Consulta cadastrada com sucesso.
- `400 Bad Request`: Falha na validação (e.g., data no passado, formato incorreto).
- `409 Conflict`: Conflito de horário de agendamento (Assumindo que a validação de `RF18` seja implementada).

6. GET /view/patients - Listar Pacientes

Descrição: Lista todos os pacientes cadastrados e atrelados ao usuário (dentista) logado. Esta lista não inclui os dados detalhados de odontograma e procedimentos para otimizar o carregamento.

Método HTTP: `GET`

Endpoint: `/view/patients`

Exemplo de Resposta (JSON): Retorna uma lista de objetos `Patient`.

Códigos de Resposta:

- `200 OK`: Retorna a lista de pacientes (pode ser vazia).
- `401 Unauthorized`: O usuário não está autenticado.

7. GET /view/patients/{id} - Consultar Paciente por ID

Descrição: Retorna os dados completos de um paciente específico, incluindo as observações gerais, a data de início do tratamento e o prontuário odontológico (odontograma com procedimentos aninhados).

Método HTTP: `GET`

Endpoint: `/view/patients/{id}`

Exemplo de Resposta (JSON): Retorna um objeto `Patient`.

Códigos de Resposta:

- `200 OK`: Paciente encontrado.
- `404 Not Found`: Paciente não encontrado ou não pertence ao usuário logado.

8. PUT /view/patients/update/{id} - Atualizar Dados do Paciente

Descrição: Sobrescreve as informações de um paciente existente (RF15). Permite a alteração de campos como nome, CPF, telefone, datas e observações.

Método HTTP: `PUT`

Endpoint: `/view/patients/update/{id}`

Corpo da Requisição (JSON) - `Patient`: Estrutura completa do objeto `Patient` com os novos dados.

Códigos de Resposta:

- `200 OK`: Paciente atualizado com sucesso.

- `400 Bad Request`: Falha na validação dos dados (e.g., CPF inválido, datas futuras).
- `404 Not Found`: Paciente não encontrado.

9. GET /view/consultations - Listar Consultas

Descrição: Lista todos os agendamentos de consultas associados ao usuário logado.

Método HTTP: `GET`

Endpoint: `/view/consultations`

Exemplo de Resposta (JSON): Retorna uma lista de objetos `Consultation`.

Códigos de Resposta:

- `200 OK`: Retorna a lista de consultas.

10. PUT /view/consultations/update/{id} - Alterar Consulta

Descrição: Sobrescreve as informações de uma consulta agendada (RF19). Permite mudar o dia, horário, paciente e a flag de lembrete do WhatsApp (RF17). O sistema deve verificar conflitos de horário.

Método HTTP: `PUT`

Endpoint: `/view/consultations/update/{id}`

Corpo da Requisição (JSON) - `Consultation`: Estrutura completa do objeto `Consultation` com os novos dados.

Códigos de Resposta:

- `200 OK`: Consulta atualizada com sucesso.
- `400 Bad Request`: Falha na validação (data no passado, formato incorreto).
- `409 Conflict`: Conflito de horário de agendamento (Assumindo que a validação de `RF18` seja implementada).
- `404 Not Found`: Consulta não encontrada.

11. POST /view/tooth-procedures/patient/{patientId} - Cadastrar Procedimento Odontológico

Descrição: Registra um novo procedimento no odontograma do paciente especificado. O dente é identificado pelo `toothNumber` (11 a 48) e o sistema atribui o `toothName` correspondente automaticamente.

Método HTTP: `POST`

Endpoint: `/view/tooth-procedures/patient/{patientId}`

Corpo da Requisição (JSON) - `ToothProcedure` (Dados necessários para o cadastro):

```
{
  "toothNumber": 16,
  "procedureName": "Restauração",
  "description": "Restauração em resina do 1º Molar Superior Direito.",
  "procedureDate": "2025-12-14"
}
```

Códigos de Resposta:

- `200 OK`: Procedimento cadastrado com sucesso.
- `400 Bad Request`: Dados inválidos (e.g., `toothNumber` fora do intervalo, campos obrigatórios vazios).
- `404 Not Found`: Paciente não encontrado.

12. GET /view/tooth-procedures/patient/{patientId} - Listar Procedimentos de um Paciente

Descrição: Retorna todos os procedimentos odontológicos (`ToothProcedure`) registrados para um paciente específico.

Método HTTP: `GET`

Endpoint: `/view/tooth-procedures/patient/{patientId}`

Exemplo de Resposta (JSON): Retorna uma lista de objetos `ToothProcedure`.

Códigos de Resposta:

- `200 OK`: Retorna a lista de procedimentos (pode ser vazia).
- `404 Not Found`: Paciente não encontrado.

13. DELETE /view/tooth-procedures/{id} - Excluir Procedimento Odontológico

Descrição: Exclui um registro de procedimento odontológico do histórico do paciente.

Método HTTP: `DELETE`

Endpoint: `/view/tooth-procedures/{id}`

Códigos de Resposta:

- `200 OK`: Procedimento excluído com sucesso.
- `404 Not Found`: Procedimento não encontrado.
- `403 Forbidden`: Usuário não tem permissão para excluir.

14. GET /view/home/stats - Estatísticas do Painel

Descrição: Retorna as principais métricas de resumo para o painel de controle (dashboard) do usuário logado, como o total de pacientes, o total de consultas agendadas e o número de consultas agendadas para o dia atual.

Método HTTP: `GET`

Endpoint: `/view/home/stats`

Exemplo de Resposta (JSON):

```
{
  "totalPatients": 45,
  "totalConsultations": 120,
  "todayConsultations": 5
}
```

Códigos de Resposta:

- `200 OK`: Retorna o mapa de estatísticas.
- `500 Internal Server Error`: Erro ao buscar os dados nas camadas de serviço.

15. GET /view/home/consultations/today - Consultas do Dia

Descrição: Retorna uma lista detalhada de todas as consultas que estão agendadas para o dia atual do usuário logado.

Método HTTP: `GET`

Endpoint: `/view/home/consultations/today`

Exemplo de Resposta (JSON): Retorna um mapa contendo o campo `"consultations"` com uma lista de objetos `Consultation`.

Códigos de Resposta:

- `200 OK`: Retorna a lista de consultas do dia.
- `500 Internal Server Error`: Erro ao processar a busca.

16. POST /view/user/getRecoveryPasswordCode - Enviar Código de Recuperação

Descrição: Inicia o processo de recuperação de senha. Recebe o e-mail do usuário, verifica se o e-mail existe e, em caso positivo, gera um código de recuperação e o envia para o endereço de e-mail por SMTP (conforme configurado).

Método HTTP: `POST`

Endpoint: `/view/user/getRecoveryPasswordCode`

Corpo da Requisição (JSON) - `PasswordRecovery` (apenas o e-mail):

```
{
  "email": "dentista@exemplo.com"
}
```

Códigos de Resposta:

- `200 OK`: Código gerado e enviado com sucesso. Retorna o código no corpo para fins de desenvolvimento/teste.
- `404 Not Found`: E-mail não cadastrado no sistema.
- `400 Bad Request`: Erro no envio do e-mail.