



igti

# RELATÓRIO

---

## PROJETO APLICADO

Instituto de Gestão e Tecnologia da Informação  
Relatório do Projeto Aplicado

# App de Gerenciamento para Psicólogos

Marcelo José Aragão Ramos

Orientador(a): Professor Bruno Augusto Teixeira

03/06/2022



**MARCELO JOSÉ ARAGÃO RAMOS**

**INSTITUTO DE GESTÃO E TECNOLOGIA DA INFORMAÇÃO**

**RELATÓRIO DO PROJETO APLICADO**

# **App de Gerenciamento para Psicólogos**

Relatório de Projeto Aplicado  
desenvolvido para fins de conclusão do  
curso MBA em Desenvolvimento Full  
Stack.

Orientador (a): Bruno Augusto Teixeira

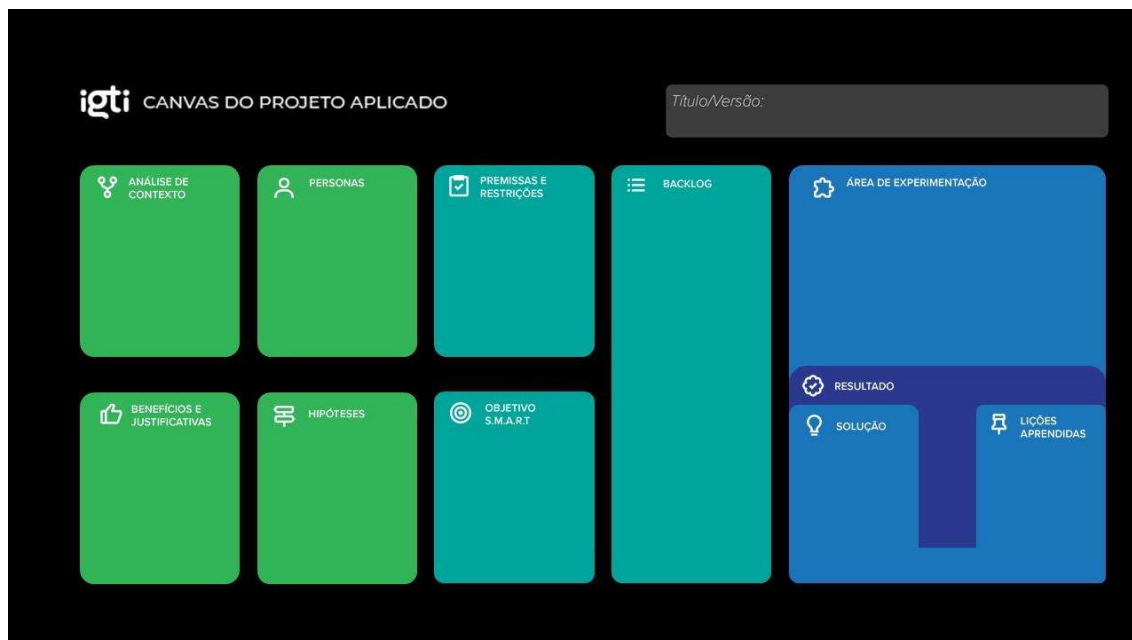
**Fortaleza  
09/06/2022**

# Sumário

1. CANVAS do Projeto Aplicado	4
1.1 Desafio	5
1.1.1 Análise de Contexto	5
1.1.2 Personas	7
1.1.3 Benefícios e Justificativas	8
1.1.4 Hipóteses	9
1.2 Solução	10
1.2.1 Objetivo SMART	10
1.2.2 Premissas e Restrições	10
1.2.3 Backlog de Produto	11
2. Área de Experimentação	12
2.1 Sprint 1	12
2.1.1 Solução	12
• Evidência do planejamento:	12
• Evidência da execução de cada requisito:	13
• Evidência dos resultados:	17
2.1.2 Experiências vivenciadas	18
2.2 Sprint 2	20
2.2.1 Solução	20
• Evidência do planejamento:	20
• Evidência da execução de cada requisito:	20
• Evidência dos resultados:	20
2.2.2 Experiências vivenciadas	23
2.3 Sprint 3	24
2.3.1 Solução	24
• Evidência do planejamento:	24
• Evidência da execução de cada requisito:	24
• Evidência dos resultados:	27
2.3.2 Experiências vivenciadas	28
3. Considerações Finais	29
3.1 Resultados	29
3.2 Contribuições	29
3.3 Próximos passos	30

## 1. CANVAS do Projeto Aplicado

Figura conceitual, que representa todas as etapas do Projeto Aplicado.



## 1.1 Desafio

### 1.1.1 Análise de Contexto

Com a chegada do COVID-19 e a população tendo de ficar mais tempo em suas casas por conta do lockdown mundial, acabou trazendo à tona muitos problemas psicológicos para diversas pessoas, como ansiedades, depressões dentre outros.

A figura do psicólogo nessa fase é de essencial necessidade para todos que precisam da ajuda. Porém muitos desses profissionais acabaram também tendo que trabalhar de suas próprias casas em formato de home-office, fazendo com que as organizações de sessões e controle financeiro ficassem a cargo deles.

Muitos acabavam fazendo o uso de aplicativos ou de planilhas, porém sem integração e sem unificação nenhuma. Exemplo de controle de sessão é o que demonstra a Figura 1, onde tem o aplicativo do Google Calendar.

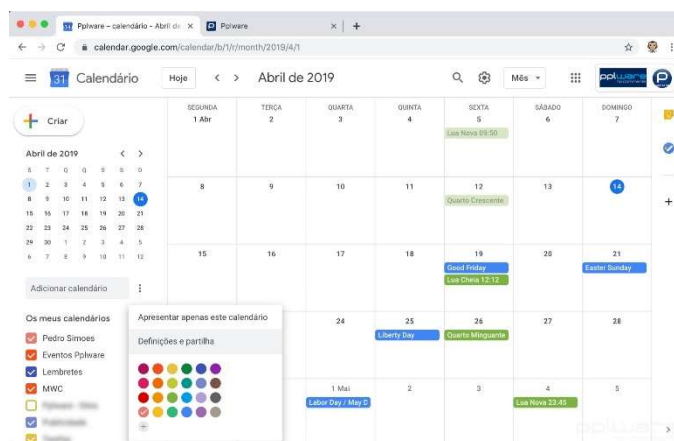


Figura 1 - Aplicativo Google Calendar

Exemplo de controle financeiro ficava a cargo de planilhas em excel, conforme a Figura 2.

Data	Hora	Nome do Paciente	Valor da sessão
01/05/2022	10:00	Paciente 1	R\$ 100,00
01/05/2022	11:00	Paciente 2	R\$ 70,00
01/05/2022	14:00	Paciente 3	R\$ 100,00
01/05/2022	15:00	Paciente 4	R\$ 150,00
01/05/2022	16:00	Paciente 5	R\$ 30,00
02/05/2022	09:00	Paciente 6	R\$ 30,00
02/05/2022	10:00	Paciente 7	R\$ 150,00
02/05/2022	11:00	Paciente 8	R\$ 100,00
03/05/2022	10:00	Paciente 9	R\$ 100,00

Figura 2 - Planilha do Excel

Analisando o formato de trabalho e controle desses profissionais, percebe-se uma grande necessidade de controle e gerenciamento de suas tarefas.

Com essas informações bem analisadas e verificadas, é possível comprovar a dor e a necessidade desses psicólogos, como: onde posso verificar as sessões que terei naquele dia? Qual foi o meu recebimento financeiro dentro de um período específico? Dentro das sessões do mês, quais os pacientes já efetuaram pagamentos?

As ilustrações abaixo irão ajudar no melhor no entendimento dessas dores: Na figura 3 podemos ver a matriz CSD. Onde podemos destacar suas certezas, suposições e dúvidas perante o sistema.

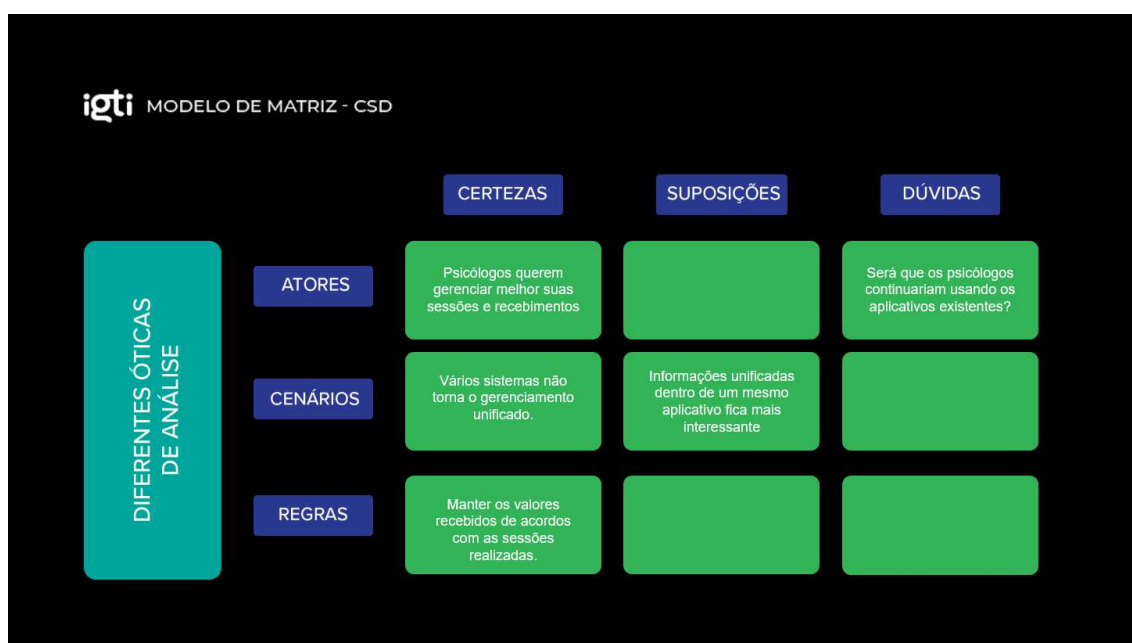


Figura 3 - Matriz CSD

Com a matriz CSD exposta acima, pode-se verificar a real necessidade passada por esses profissionais.

Entrando em imersão profunda e se colocando no lugar deles, é montando o POEMS:

- Pessoas
- Objetos
- Ambiente
- Mensagem
- Serviços

Como demonstrado na figura 4, que mostra a imersão profunda relevante ao sistema.

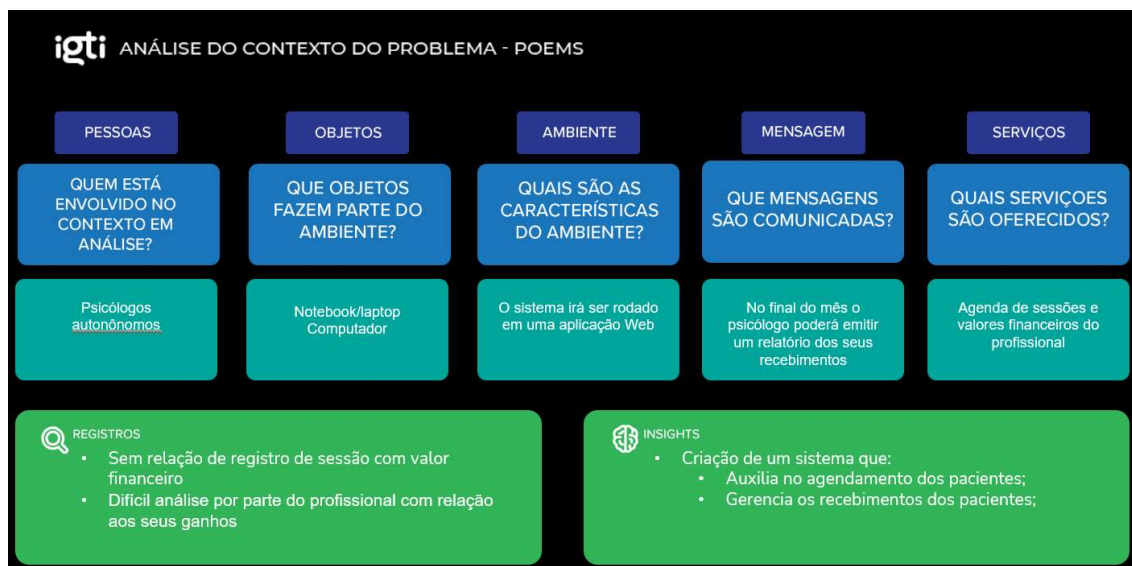


Figura 4 - Observação POEMS

## 1.1.2 Personas

Os grandes interessados pela resolução desse sistema são os psicólogos, principalmente os autônomos que não fazem uso de secretárias precisam realizar suas atividades de gerenciamento de sessões e financeiros sozinhos.

Verônica é uma psicóloga com idade entre 41-59 anos. Sua responsabilidade profissional é de tratar pacientes com problemas emocionais, e tem em seu objetivo tratá-los através da resposta emocional do paciente. O seu grande desafio é não ter uma ferramenta que possa auxiliar no seu gerenciamento de sessões/recebíveis.

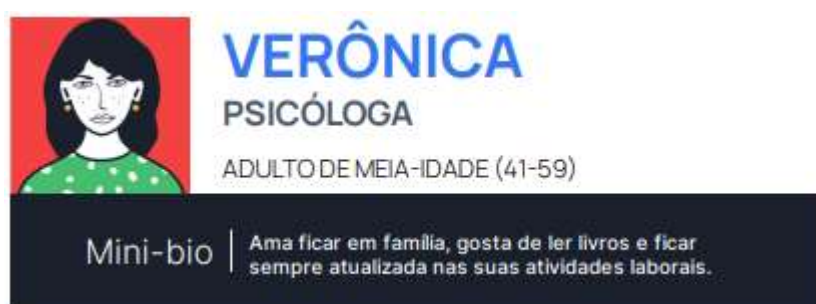


Figura 5 - Persona



A figura 6 ilustra a persona principal dentro desse aplicativo.

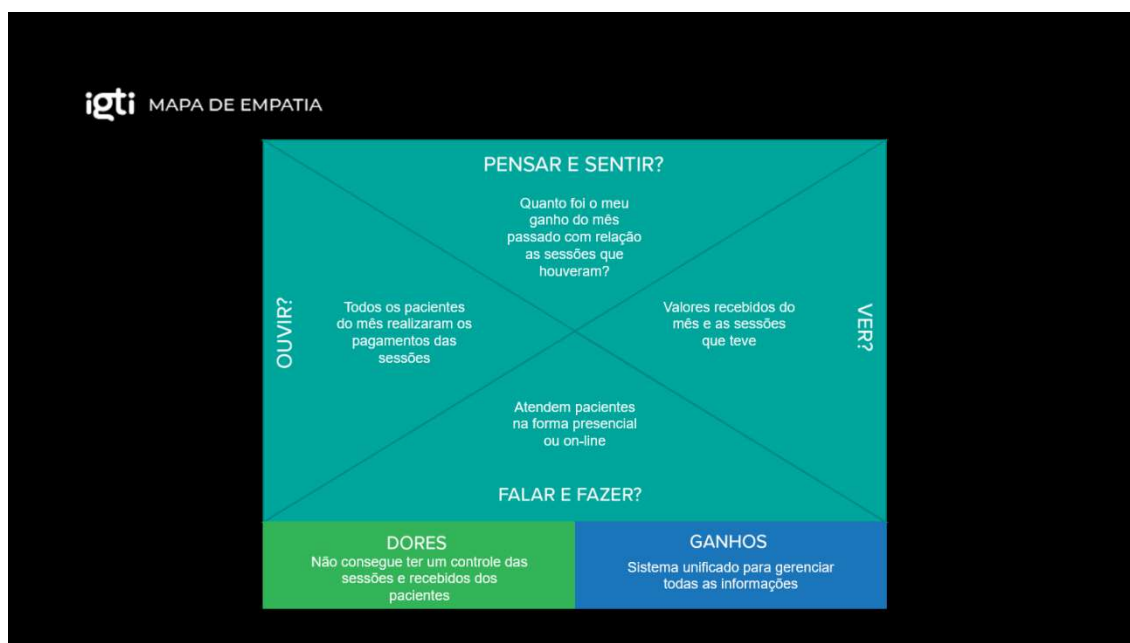


Figura 6 - Mapa de Empatia

### 1.1.3 Benefícios e Justificativas

Nesta seção pode-se observar através da tabela 1 que corresponde a ferramenta de Blueprint, o mapeamento e a listagem das ações do cliente, para resolver o problema em análise.

Ações do cliente	Gerenciar Sessões	Gerenciar Valores
<b>Objetivos</b>	Verificar como está ou como foi as suas sessões em um determinado dia/período	Saber como está seus recebimentos dos pacientes
<b>Atividades</b>	Cadastrar as sessões dos pacientes	Manter os valores das sessões
<b>Questões</b>	Irei ter pacientes para atender hoje?	O paciente realizou o pagamento?
<b>Barreiras</b>	Não se aplica	Não se aplica
<b>Funcionalidades</b>	Informar qual paciente será atendidos nas sessões em questão.	Informar o valor que o paciente pagou pela sessão
<b>Interação</b>	Acessar o sistema e informar as sessões	Acessar o sistema e informar os valores
<b>Onde ocorre</b>	Qualquer ambiente que tenha internet	Qualquer ambiente que tenha internet
<b>Tarefas aparentes</b>	Não permitir colocar dois pacientes em uma mesma sessão.	Não se aplica
<b>Tarefas escondidas</b>	Implementação do sistema com Node.js e React	Implementação do sistema com Node.js e React
<b>Processos de suporte</b>	Mostrar os ganhos que o sistema irá ter para o psicólogo	Mostrar os ganhos que o sistema irá ter para o psicólogo
<b>Saída desejável</b>	Manter o psicólogo informado de suas sessões	Manter o psicólogo informado dos pagamentos de seus pacientes

Tabela 1 - Blueprint

A ferramenta ilustrada na figura 7 mostra o Canvas Proposta de Valor, onde podemos entender o cliente e verificar como o produto irá se adequar a suas necessidades.

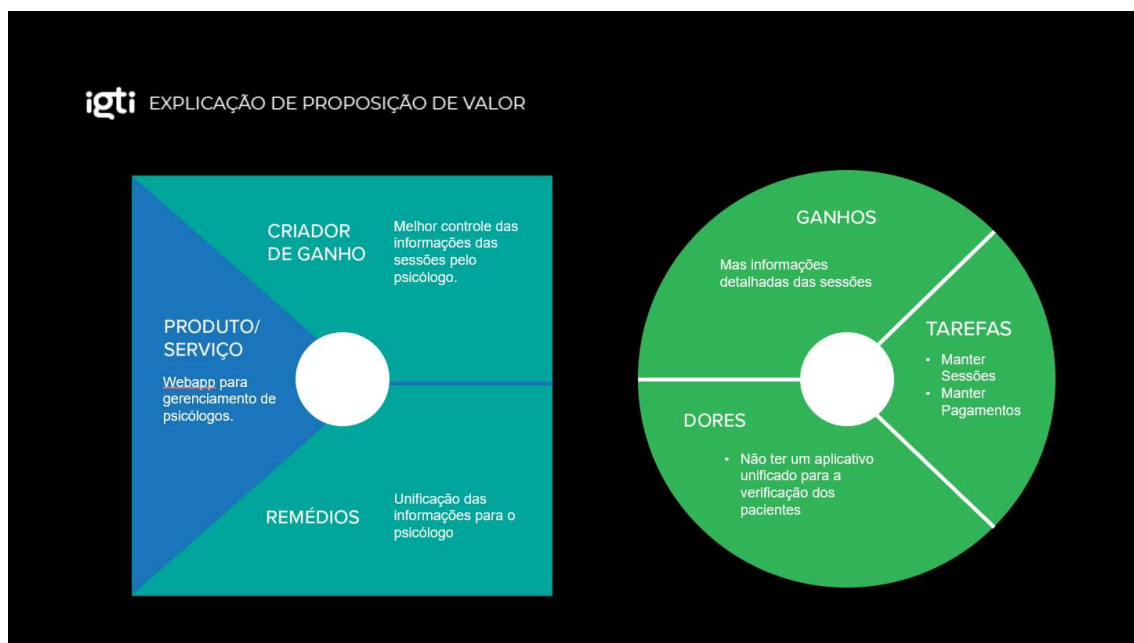


Figura 7 - Canvas da Proposta de Valor

#### 1.1.4 Hipóteses

Está relacionado abaixo algumas hipóteses com respeito ao problema da falta do gerenciamento por parte dos psicólogos.

- Psicólogos não conseguem ter o controle de suas sessões e os valores de recebimento por parte dos pacientes.
- Os psicólogos não sabem se a sessão ou o pagamento foi realizada ou não.

Com as hipóteses evidenciadas acima, segue algumas ideias para resolvê-las:

I1. Criação de um sistema de gerenciamento para psicólogos.

I2. Alterar a planilha para que possa ser informado todas as necessidades dos psicólogos.

Com as ideias acima expostas foi montado uma Matriz de Priorização de Ideias, conforme é possível analisar na tabela 2:

Ideias	Critérios de Comparação						Somatório
	B	A	S	I	C	O	
I1	5	4	4	4	2	5	24
I2	2	3	2	1	5	5	18

*Tabela 2 - Matriz de priorização de ideias*

A utilização para a construção dessa matriz foi a base da matriz BASICO:

- Benefícios
- Abrangências
- Satisfação
- Investimentos
- Cliente
- Operacionalidade

## 1.2 Solução

### 1.2.1 Objetivo SMART

Desenvolver um aplicativo em até 2 meses, com cerca de 2 horas por dia, um MVP para gerenciar as sessões dos psicólogos.

### 1.2.2 Premissas e Restrições

As premissas vistas para este projeto são:

- Utilização de 2 horas por dia para o desenvolvimento do mesmo.
- Não será desenvolvido em primeiro momento a funcionalidade de login.

As restrições encontradas para o mesmo são:

- O sistema será desenvolvido somente para o ambiente Web.
- O MVP deverá ser desenvolvido durante o Projeto Aplicado, que tem a duração de 2 meses.

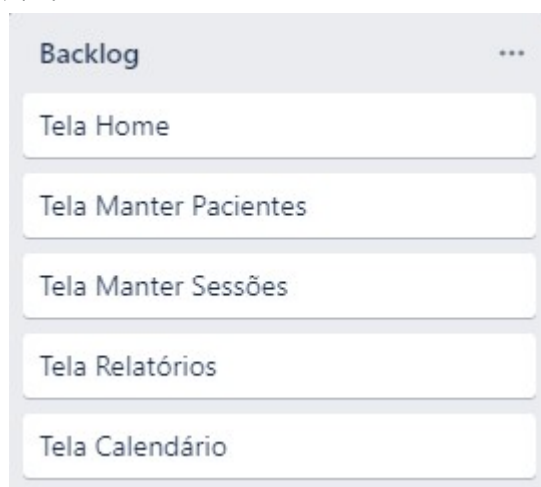
Com as informações das premissas e restrições do projeto, foi levantado alguns riscos conforme tabela 3 - Matriz de riscos.

Risco Identificado	Impacto Potencial	Ações Preventivas	Ações Corretivas
Tempo de desenvolvimento	Sistema Inutil	Focar no desenvolvimento do trabalho	Sempre prestar atenção no desenvolvimento das sprints
Aumentar o escopo	Sistema Inconsistente	Sempre olhar o backlog criado	Focar nas dores do usuário
Sistema fora do ar	Sistema Inutil	Distribuir o sistema em vários servidores	Manter o sistema mais estável possível
Aparecer outro sistema	Sistema Inutil	Sempre monitorar e atualizar o sistema	Sempre melhorar o sistema para agradar o usuário

*Tabela 3 - Matriz de riscos*

### 1.2.3 Backlog de Produto

Segue abaixo a figura 8 onde pode ser visto a proposta de backlog para o desenvolvimento do MVP:



*Figura 8 - Backlog do produto*

A informação do backlog pode ser visto e tratado também no link: <https://trello.com/b/k8yt77JU/projeto-aplicado>

## 2. Área de Experimentação

### 2.1 Sprint 1

As principais funcionalidades implementadas dentro da Sprint 1 foram as mais básicas para consistir no MVP.

As tecnologias utilizadas para a construção do sistema do projeto aplicado foram:

- Front End: React.js
- Back End: Node.js

O foco da Sprint foi desenvolver tanto o front end quanto o back end, ou seja, seguir o fluxo total das telas desde a visualização até a sua inserção nas tabelas correspondentes. Foram utilizadas várias tecnologias que serviu para auxiliar no desenvolvimento, como descrito abaixo:

- Front End
  - Tailwind-Css - Estilização das páginas;
  - React-Table - Criação de tabelas;
  - React-Router-Dom - Roteamento das páginas;
  - Fontawesome - Inserção de ícones nas páginas;
  - Material-ui - Utilização de componentes nas páginas;
- Back End
  - Sequelize - Auxilia na conexão com o banco de dados
  - Express - Roteamento das api's

#### 2.1.1 Solução

- **Evidência do planejamento:**

Para a Sprint 1 foram desenvolvidas as seguintes telas, conforme visto na figura 9 que mostra a tela do trello:

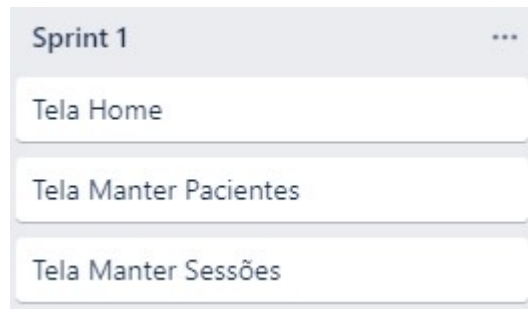


Figura 9 - Visão da Sprint 1 no Trello

- Evidência da execução de cada requisito:

#### Tarefa Tela Home

Utilizando a biblioteca react-router-dom, foi criado a rota para a tela de Home, como podemos ver na figura 10.

```
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<GerenciamentoPsicoScreen />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Figura 10 - Function App

Já na figura 11 pode ser observado o componente de onde ficará todos os links para acessar cada tela específica.

```
return (
  <>
    <Header title="Gerenciamento de Psicólogos" />
    <AppBar position="static">
      <Tabs value={value} onChange={handleChange}>
        <Tab label="Home" icon={<i className="fa-solid fa-house" />} />
        <Tab label="Paciente" icon={<i className="fa-solid fa-user" />} />
        <Tab label="Sessões" icon={<i className="fa-solid fa-file" />} />
      </Tabs>
    </AppBar>
    <TabPanel value={value} index={0}>
      <HomePage />
    </TabPanel>
    <TabPanel value={value} index={1}>
      <Paciente />
    </TabPanel>
    <TabPanel value={value} index={2}>
      <Sessao />
    </TabPanel>
  </>
)
```

Figura 11 - GerenciamentoPsicoScreen

Na figura 12 podemos ver o componente que levará para a tela inicial, a tela HomePage.

```
return (
  <div className="flex flex-col space-y-2 p-4">
    <h3 className="text-center mb-6">Bem vindo ao Sistema de Gerenciamento de Psicólogo</h3>
    <ul className="text-center mb-6">
      <li>
        Acesse Pacientes para cadastrar seus pacientes.
      </li>
      <li>
        Acesse Sessões para cadastrar suas sessões.
      </li>
    </ul>
  </div>
)
```

Figura 12 - HomePage

### Tarefa Tela Manter Pacientes

Utilizando o express no lado do back end pode ser observado na figura 13 as rotas utilizadas dentro da aplicação.



```
router.post('/', PacienteController.createPaciente);
router.put('/', PacienteController.updatePaciente);
router.delete('/:id', PacienteController.deletePaciente);
router.get('/', PacienteController.getPacientes);
router.get('/:id', PacienteController.getPaciente);
```

Figura 13 - Rotas da API do paciente

Na figura 14 pode-se verificar como ficou interessante a utilização da biblioteca react-table que ajudou na visualização dos pacientes cadastrados.

```
<CustomTable
  columns={columns}
  data={pacientes.map((paciente) => ({
    ...paciente,
    acoes: (
      <div>
        <button
          type="button"
          onClick={() => deletar(paciente.pacienteId)}
        >
          <i className="text-center" aria-hidden="true" title="Deletar">
            <AiFillDelete />
          </i>
        </button>
        <button
          type="button"
          onClick={() => atualizar(paciente)}
        >
          <i className="text-center" aria-hidden="true" title="Atualizar">
            <AiFillEdit />
          </i>
        </button>
      </div>
    )
  })
)}>
```

Figura 14 - Tela para mostrar a lista de pacientes

### Tarefa Manter Sessões

Para a tarefa de manter sessão pode-se ver na figura 15 ver um trecho de código onde é realizado a chamada ao banco de dados para realizar suas devidas tarefas.



```

async function insertSessao(sessao) {
  try {
    return await Sessao.create(sessao);
  } catch (error) {
    throw error;
  }
}

async function updateSessao(sessao) {
  try {
    await Sessao.update(
      {
        valor: sessao.valor,
        observacao: sessao.observacao
      },
      {
        where: {
          sessaoId: sessao.sessaoId,
        },
      }
    );
    return await getSessao(sessao.sessaoId);
  } catch (error) {
    throw error;
  }
}

```

Figura 15 - Repositório da sessão

Na figura 16 pode ser visto o componente CustomTable que também é usado na tela para mostrar as sessões.

```
return (
  <table {...getTableProps()}>
    <thead>
      {headerGroups.map(headerGroup => (
        <tr {...headerGroup.getHeaderGroupProps()}>
          {headerGroup.headers.map(column => (
            <th {...column.getHeaderProps()}>{column.render('Header')}</th>
          ))}
        </tr>
      ))}
    </thead>
    <tbody {...getTableBodyProps()}>
      {rows.map((row, i) => {
        prepareRow(row)
        return (
          <tr {...row.getRowProps()}>
            {row.cells.map(cell => {
              return <td {...cell.getCellProps()}>{cell.render('Cell')}</td>
            })}
          </tr>
        )
      })}
    </tbody>
  </table>
)
```

Figura 16 - Componente CustomTable

- Evidência dos resultados:

#### Tarefa Tela Home

Na figura 17 é possível ver a renderização da tela principal do sistema.

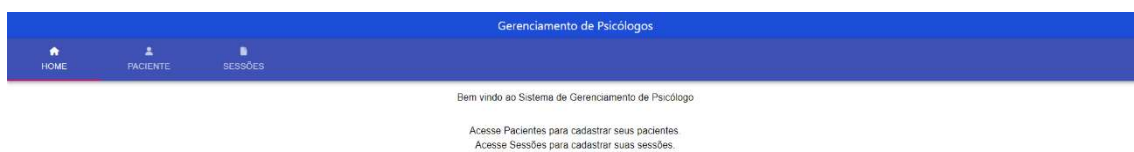


Figura 17 - Renderização da Tela Inicial

Para descrever as tabelas que foram utilizadas na estrutura de dados dentro da aplicação foi realizado o seguinte comando SQL, conforme demonstrado na figura 18.

```
CREATE TABLE pacientes(
paciente_id SERIAL PRIMARY KEY,
nome VARCHAR NOT NULL,
email VARCHAR NOT NULL,
telefone VARCHAR NOT NULL,
endereço VARCHAR NOT NULL
);

CREATE TABLE sessoes(
sessao_id SERIAL PRIMARY KEY,
paciente_id INT NOT NULL,
data TIMESTAMP NOT NULL,
observacao VARCHAR NOT NULL,
valor NUMERIC NOT NULL,
CONSTRAINT fk_pacientes FOREIGN KEY (paciente_id) REFERENCES pacientes (paciente_id)
);
```

Figura 18 - Tabelas usadas no banco de dados

## Tarefa Manter pacientes

Na figura 19 é evidenciado a tela renderizada da tela de paciente

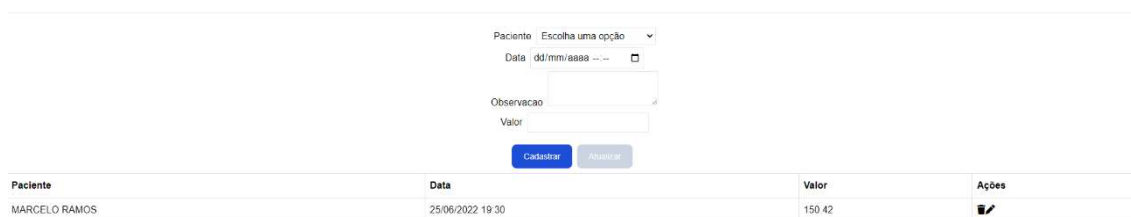


Nome	E-mail	Telefone	Endereço	Ações
MARCELO RAMOS	mjaramos@gmail.com	991790249	AP 1103	 
MARCELO J A RAMOS	mjaramos@gmail.com	7777	RUA HENRIQUETA GALENO, 380, APTO 1103	 

Figura 19 - Tela de Paciente

## Tarefa Manter Sessão

Na figura 20 pode ser também verificado a evidência da tela de sessão.





Paciente	Data	Valor	Ações
MARCELO RAMOS	25/06/2022 19:30	150.42	 

Figura 20 - Tela de Sessão

## 2.1.2 Experiências vivenciadas

Durante a Sprint 1 foram constatadas algumas experiências que serviu de lição, como:

- As inúmeras bibliotecas disponibilizadas na internet para ajudar e auxiliar na construção de telas foi de grande importância;

- A construção do back end no início do projeto, foi interessante por tratar-se de um assunto que é mais fácil de assimilar e compreender;

## 2.2 Sprint 2

O foco da Sprint 2 foi nos relatórios em que o usuário irá poder tirar do sistema. Para a apresentação e visualização desses relatórios, os pacientes e sessões devem de estar previamente cadastrados, conforme visto e avaliado na Sprint 1.

Para esta Sprint foi utilizada apenas uma tecnologia para a geração dos relatórios em pdf:

- PDFMake

### 2.2.1 Solução

- Evidência do planejamento:

Conforme evidenciado no backlog do produto a tarefa a ser realizada na Sprint 2 está sendo demonstrada na figura 21:

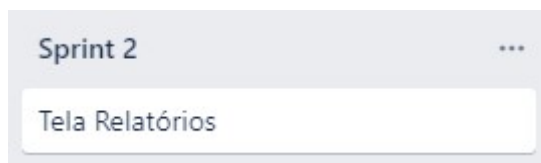


Figura 21 - Visão da Sprint 2 no Trello

- Evidência da execução de cada requisito:

#### Tarefa Tela Relatório

A chamada dos relatórios utilizando o PDFMake facilitou muito o trabalho para a sua geração. No exemplo da figura 22 pode-se observar que a aplicação chama apenas o componente do relatório:

```
const reportPaciente = async (relatorio: RelatorioState) => {
  const pacientes = await apiGetPacientesLikeNome(relatorio.nome);
  PacienteReports(pacientes);
}
```

Figura 22 - Chamada do relatório de pacientes

A configuração do relatório é realizada através do componente criado para cada um. Na figura 23 é demonstrado a parte inicial da chamada do relatório, nela é possível ser visto a definição do título e dos dados:

```
export default function PacienteReports(pacientes) {
  pdfMake.vfs = pdfFonts.pdfMake.vfs;

  const reportTitle = [
    {
      text: 'Pacientes',
      fontSize: 15,
      bold: true,
      margin: [15, 20, 0, 45]
    }
  ]

  const dados = pacientes.map((paciente) => {
    return [
      { text: paciente.nome, fontSize: 9, margin: [0, 2, 0, 2] },
      { text: paciente.email, fontSize: 9, margin: [0, 2, 0, 2] },
      { text: paciente.telefone, fontSize: 9, margin: [0, 2, 0, 2] },
      { text: paciente.endereco, fontSize: 9, margin: [0, 2, 0, 2] },
    ]
  })
}
```

Figura 23 - Parte Inicial do Componente

Na figura 24 é possível observar os detalhes do relatório, onde consegue ser visualizado o nome de cada coluna, como também os dados que serão impressos, já vistos na figura passada:

```
const details = [
  {
    table: {
      headerRows: 1,
      widths: ['*', '*', '*', '*'],
      body: [
        [
          { text: 'Nome', style: 'tableHeader', fontSize: 10 },
          { text: 'E-mail', style: 'tableHeader', fontSize: 10 },
          { text: 'Telefone', style: 'tableHeader', fontSize: 10 },
          { text: 'Endereço', style: 'tableHeader', fontSize: 10 },
        ],
        ...dados
      ]
    },
    layout: 'lightHorizontalLines'
  }
]
```

Figura 24 - Detalhes do Componente do Relatório



Finalizando o componente é possível visualizar na figura 25 o rodapé, as definições e a criação do PDF.

```
function Rodape(currentPage, pageCount){
  return [
    {
      text: currentPage + ' / ' + pageCount,
      alignment: 'right',
      fontSize: 9,
      margin: [0, 10, 20, 0]
    }
  ]
}

const pacienteDefinitions = {
  pageSize: 'A4',
  pageMargins: [15, 50, 15, 40],
  header: [reportTitle],
  content: [details],
  footer: Rodape
}

pdfMake.createPdf(pacienteDefinitions).download();
```

Figura 25 - Detalhes final do componente do Relatório

- Evidência dos resultados:

#### Tarefa Relatório

Na figura 26 é possível observar o relatório dos pacientes:

#### Pacientes

Nome	E-mail	Telefone	Endereço
Paciente 1	paciente1@email.com	(99) 99999-9999	Rua Paciente 1, número 1
Paciente 2	paciente2@email.com	(88) 98888-8888	Rua Paciente 2, nº 2
Paciente 3	paciente3@email.com	(77) 97777-7777	Rua Paciente 3, s/n

Figura 26 - Relatório de Pacientes

Na figura 27 já é possível visualizar o relatório de sessões:

#### Sessões

Paciente	Data	Valor
Paciente 1	03/01/2022 09:00	150.00
Paciente 2	03/01/2022 10:00	150.00
Paciente 3	03/01/2022 11:00	75.00
Paciente 1	14/02/2022 09:00	150.00
Paciente 2	21/02/2022 09:00	150.00
Paciente 3	04/07/2022 10:00	75.00

*Figura 27 - Relatório de Sessões*

### 2.2.2 Experiências vivenciadas

Durante a Sprint 2, por mais que tenha sido desenvolvido apenas uma única tarefa a mesma foi a mais árdua. A falta de experiência com a tecnologia para a geração de relatório fez com que tivesse que realizar diversas buscas na internet para encontrar a que fosse mais adaptável ao projeto.

No primeiro momento foi encontrada uma tecnologia que acabou sendo necessário baixar outras bibliotecas para que ela funcionasse, mesmo assim a aplicação apresentou diversos problemas.

Pesquisando mais sobre o assunto foi encontrado uma melhor biblioteca, onde a mesma, tinha exemplos em diversos sites dedicados a tecnologia, bem como em sites de compartilhamento de vídeos.



## 2.3 Sprint 3

O foco da Sprint 3 foi na tela em que o usuário pode verificar através de um calendário, suas sessões em que foram cadastradas. Nela é possível a verificação também se o paciente realizou o pagamento ou não.

Para esta Sprint a tecnologia utilizada para mostrar o calendário foi:

- rsuitejs

### 2.3.1 Solução

- Evidência do planejamento:

Conforme visto no backlog do produto a tarefa a ser realizada na Sprint 3 está sendo demonstrada na figura 28:

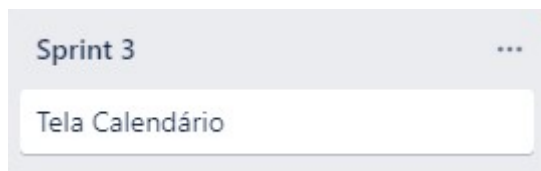


Figura 28 - Visão da Sprint 3 no Trello

- Evidência da execução de cada requisito:

#### Tarefa Tela Calendário

Continuando na utilização de bibliotecas para auxiliar na construção do MVP o rsuitejs foi de grande importância.

Apenas com um simples componente é possível a criação do calendário, como pode ser observado na figura 29:

```
<Calendar renderCell={renderCell} bordered={true} />
```

Figura 29 - Componente Calendário do rsuitejs

A propriedade principal deste componente encontra-se no renderCell, onde nela será populada cada dia em que a sessão foi cadastrada.

Antes de começar a popular o calendário se faz necessário recuperar todas as sessões estando elas ordenadas pela data da sessão.

A ordenação foi realizada no backend da aplicação, conforme é visto na figura 30

```

async function getSessoes() {
  try {
    return await Sessao.findAll({
      include: [
        {
          model: Paciente
        }
      ],
      order: [
        ['data', 'ASC']
      ],
    });
  } catch (error) {
    throw error;
  }
}

```

Figura 30 - Função de retorno das sessões ordenada por data

Com as sessões recebidas através da API, é utilizado o filtro, ou seja, a separação das sessões por data.

Na figura 31, é possível verificar a chamada da API e logo em seguida a função em que separa as datas.

```
useEffect(() => {

  async function getSessoes() {
    const bkSessoes = await apiGetSessoes()
    setSessoes(bkSessoes);
  }

  getSessoes();

}, [])

function getTodoList(date) {
  const day = getDate(date);
  const month = getMonth(date) + 1;
  return sessoes.filter((sessao) => {
    let dia = parseInt(moment(sessao.data).format('DD'))
    let mes = parseInt(moment(sessao.data).format('MM'))
    return day === dia && month === mes
  })
}
```

Figura 31 - Recuperação das sessões e função para separar

Com as informações das sessões populadas a função de renderCell começa a ser construído.

Será possível verificar mais dois componentes utilizados, que também se encontram na biblioteca rsuitejs, são elas Whisper e Popover. As duas irão trabalhar em conjunto para quando no mesmo dia houver mais de uma sessão cadastrada. A figura 32 pode ser demonstrada como fica o código implementado.

```
function renderCell(date) {
  const list = getTodoList(date);
  const displayList = list.filter((item, index) => index < 2);

  if (list.length) {
    const moreCount = list.length - displayList.length;
    const moreItem = (
      <li>
        <Whisper
          placement="top"
          trigger="click"
          speaker={
            <Popover>
              {list.map((item, index) => (
                <p key={index}>
                  <b>{moment(item.data).utc().format('HH:mm')}</b> - {item?.paciente?.nome}
                  {item.inPago ?
                    <i className="fa-solid fa-thumbs-up" text-green-500"/> : <i
                      className="fa-solid fa-thumbs-down" text-red-500"/>}
                </p>
              ))}
            </Popover>
          }
        >
        <a>{moreCount} more</a>
      </Whisper>
    </li>
  );
}
```

Figura 32 - Função renderCell

Finalizando a função renderCell é possível ver mais um componente da rsuitejs que é o Badge, que apenas será mostrado como forma de organização da sessão que será realizada na data. Na figura 33, pode ser visto o retorno da função.

```
return (
  <ul className="calendar-todo-list">
    {displayList.map((item, index) => (
      <li key={index}>
        <Badge /> <b>{moment(item.data).utc().format('HH:mm')}</b> - {item?.paciente?.nome}
        {item.inPago ?
          <i className="fa-solid fa-thumbs-up" text-green-500"/> :
          <i className="fa-solid fa-thumbs-down" text-red-500"/>}
      </li>
    ))}
    {moreCount ? moreItem : null}
  </ul>
);
```

Figura 33 - Retorno da função renderCell

- **Evidência dos resultados:**

A figura 34 é evidenciado a tela em que mostra o resultado de um mês específico com as sessões cadastradas:

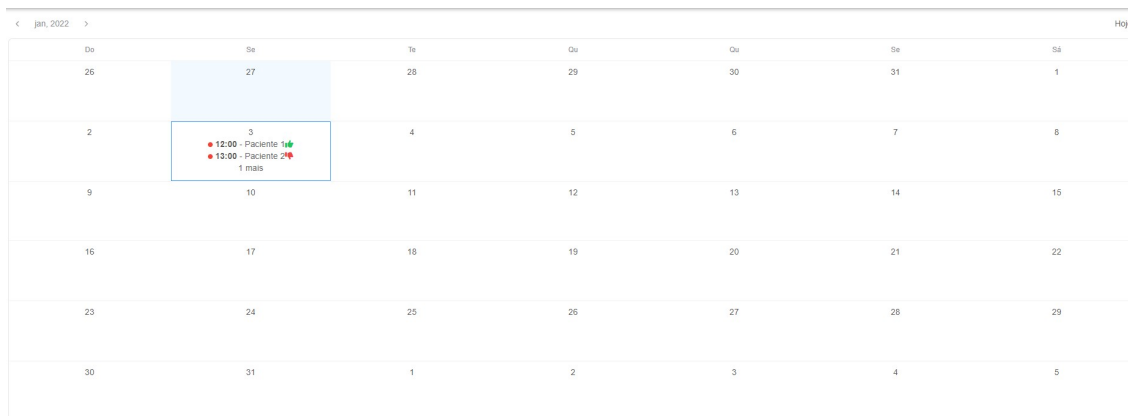


Figura 34 - Tela de Calendário

### 2.3.2 Experiências vivenciadas

No momento do desenvolvimento, dificuldades com o tempo por conta do trabalho fez com que a sprint ficasse um pouco mais estreita para seu término, mesmo assim o desenvolvimento acabou saindo um pouco da zona de conforto, o aprendizado de mais um componente que ainda não tinha sido utilizado dificultou um pouco mais a finalização da sprint.

## 3. Considerações Finais

### 3.1 Resultados

Na construção do MVP muitos resultados foram alcançados, os principais foram:

- Implementação utilizando as tecnologias aprendidas nas aulas, como: Node.js e React.
- O principal problema apresentado pelo MVP solucionado.

Com esses resultados, pode-se destacar os pontos positivos e negativos.

#### **Pontos positivos:**

- Aprendizado na tecnologia abordada, utilizando diversas bibliotecas que existem no mercado.
- Satisfação ao ver que o que foi solicitado no backlog foi desenvolvido em sua totalidade.

#### **Pontos negativos:**

- As telas desenvolvidas poderiam ter sido implementadas de forma mais organizada e responsiva.

#### **Dificuldades enfrentadas:**

- Em uma Sprint por conta do trabalho tive que me ajustar ao tempo de desenvolvimento e elaboração do documento.

#### **Experiências vivenciadas:**

- A grande quantidade de bibliotecas existentes no mercado faz com que nós tenhamos que estar sempre atualizados com as necessidades dos projetos que desenvolvemos;

### 3.2 Contribuições

O que foi desenvolvido dentro do MVP já pode ser utilizado pelos profissionais de psicologia, pois o mesmo já consegue retirar relatórios, fazendo com que o

profissional consiga visualizar os recibos dos pacientes nas sessões, como também os que ainda não foram pagos.

A visualização do calendário, mostrando as sessões dos pacientes, também tem seu destaque relevante dentro do MVP. Pois além de mostrar os dias e horas que os pacientes foram consultados ou suas futuras sessões, também é possível a visualização se a sessão já foi paga ou não.

O sistema ainda pode ser acrescido de mais funcionalidades e deixá-lo mais robusto.

### 3.3 Próximos passos

Destaco aqui o que pode ainda ser desenvolvido dentro do MVP, como:

- Implementação de autenticação/autorização - O desenvolvimento desta funcionalidade faz com que o sistema fique mais seguro e que cada profissional de psicologia possa ver somente seus pacientes e sessões
- Desenvolvimento de um layout mais organizado e responsivo - Melhorar o UI/UX para que o sistema fique mais atrativo para o usuário final;
- Implantação da aplicação nas nuvens - Realizar essa implantação faz com que a aplicação possa chegar a mais usuários, como também o mesmo mantenha a integridade de funcionamento.