

Manual de Instalação, Administração e Uso do SGSP (Sistema de Gerenciamento Sócio Pedagógico)

Versão 1.0

Autores

Érick Alexandre Felipe Sanchez
Marina Baptista dos Santos
Rafael Candido Wenceslau

Membros do projeto

Lucas Tadashi Endo
Jéssica Barbieri da Silva
Ruan Mendes da Silveira
Jônatas da Silva Buzo
Alexandre Rodrigues Medeiros

Salto, 25 de Outubro de 2016.

Sumário

1. Licença deste Documento	3
2. Introdução ao SGSP	4
2.1 Características e principais funcionalidades	4
3. Instalação do Sistema	5
3.2. Hardware mínimo necessário	5
3.3. Passo-a-passo da Instalação	5
4. Administração e Uso do Sistema	7
5. Código-fonte	9

1. Licença deste Documento

Você tem a liberdade de:



Compartilhar — copiar, distribuir e transmitir a obra.



Remixar — criar obras derivadas.

Não deixando de atribuir os devidos créditos ao autor original.



Atribuição — Você deve creditar a obra da forma especificada pelo autor ou licenciante (mas não de maneira que sugira que estes concedem qualquer aval a você ou ao seu uso da obra).



Compartilhamento pela mesma licença — Se você alterar, transformar ou criar em cima desta obra, você poderá distribuir a obra resultante apenas sob a mesma licença, ou sob uma licença similar à presente.

Ficando claro que:

Renúncia — Qualquer das condições acima pode ser renunciada se você obtiver permissão do titular dos direitos autorais.

Domínio Público — Onde a obra ou qualquer de seus elementos estiver em domínio público sob o direito aplicável, esta condição não é, de maneira alguma, afetada pela licença.

Outros Direitos — Os seguintes direitos não são, de maneira alguma, afetados pela licença:

- Limitações e exceções aos direitos autorais ou quaisquer usos livres aplicáveis;
- Os direitos morais do autor;
- Direitos que outras pessoas podem ter sobre a obra ou sobre a utilização da obra, tais como direitos de imagem ou privacidade.

Aviso — Para qualquer reutilização ou distribuição, você deve deixar claro a terceiros os termos da licença a que se encontra submetida esta obra. A melhor maneira de fazer isso é com um link para esta página.

2. Introdução ao SGSP

O SGSP é um sistema de gerenciamento sócio pedagógico que foi inteiro criado por alunos do quarto semestre do curso de **Técnico em Informática no Instituto Federal de Educação, Ciências e Tecnologia de São Paulo – Campus Salto**, onde tivemos o auxílio dos professores de Projetos de Sistemas II, **Claudio Luis Roveri Vieira e Claudio Haruo Yamamoto**. Este sistema trabalha em conjunto com mais dois sistemas, sendo eles em Desktop e Web. A ideia aqui é montar e enviar um formulário através de um dispositivo mobile Android para os outros sistemas, onde será organizado e tratado de acordo com a necessidade do mesmo.

2.1. Características e principais funcionalidades

O código fonte (que é livre e aberto) está disponível no final deste documento. . Sua implementação é em Java ME, tendo como camada de armazenamento o sistema gerenciador de banco de dados MySQL.

O sistema possui varias funcionalidades entre as quais se destacam:

- Montagem e envio de formulários necessários para prosseguir com um atendimento sociopedagógico;
- Envio instantâneo ao sistema Desktop e Web, sendo organizado no banco de dados;
- Suporte a Android 4.1 e superiores.
- Sistema simples e intuitivo, indo direto ao ponto.

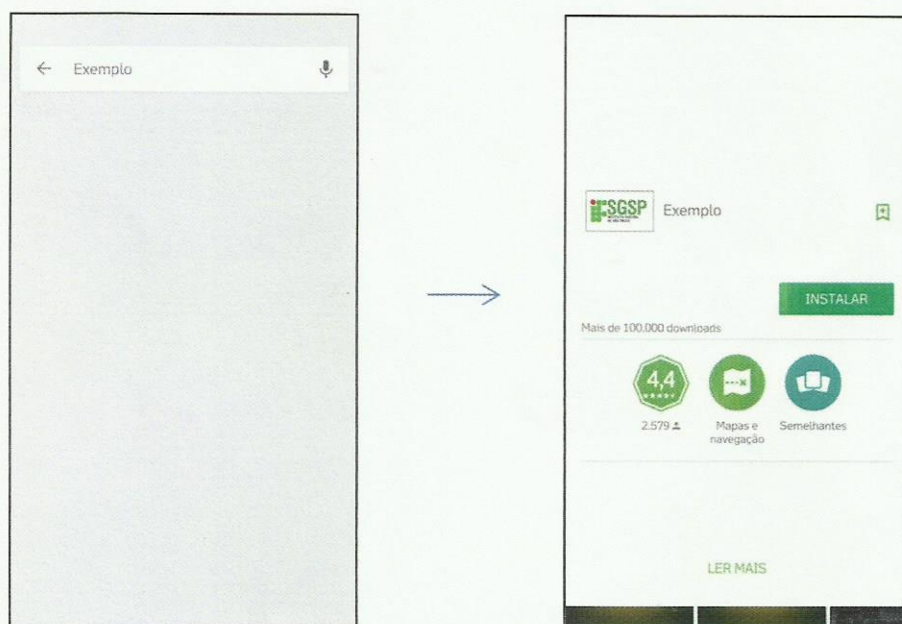
3. Instalação do Sistema

3.1. Hardware mínimo necessário

- Smartphone com sistema operacional Android 4.1 ou superior.
- Memória RAM: 512MB RAM (1GB RAM recomendável);
- HD: 5MB para instalação do aplicativo.

3.2. Passo-a-Passo para instalação

Passo 1: Baixando e instalando o aplicativo no celular



Encontre o aplicativo no Play Store (no computador ou em seu celular) e solicite o download do aplicativo.

Note que, ao pressionar o botão “instalar”, será aberta uma janela informando permissões que o aplicativo necessita.



Pressione o botão “aceitar” e aguarde o download.

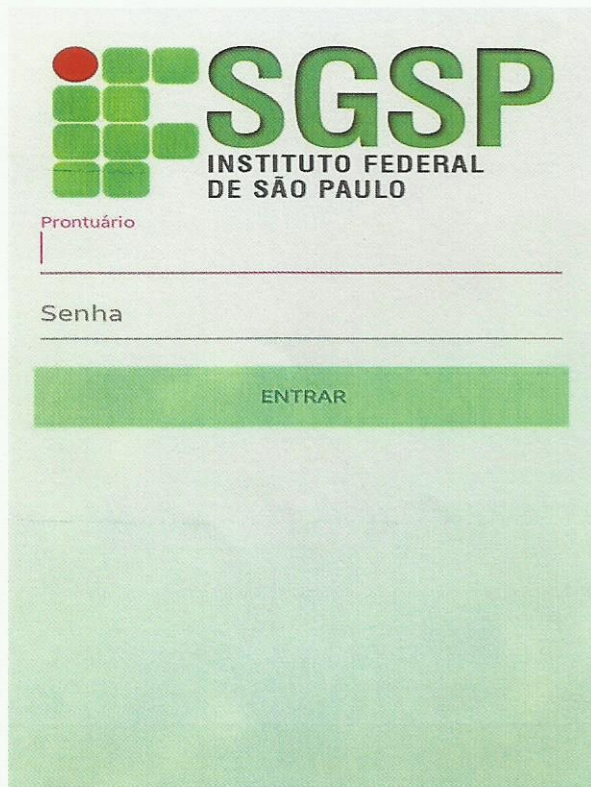
Passo 2: Ao terminar, execute o aplicativo baixado e instale-o. Caso não possua os requisitos mínimos, a instalação falhará.



Passo 3: Observe que tudo foi concluído com sucesso e em seguida abra o programa.

4. Administração e Uso do Sistema

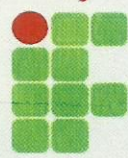
Ao abrir o aplicativo em seu celular, deve prosseguir para a tela de carregamento. O tempo de carregamento varia de acordo com o modelo e processador do smartphone, além de sua memória RAM.

A tela de login do aplicativo SGSP possui o mesmo logo do Instituto Federal de São Paulo no topo. Abaixo do logo, há dois campos de entrada: o primeiro é rotulado "Prontuário" e o segundo "Senha". Ambos os campos possuem uma linha horizontal para digitação. Abaixo dos campos, há um botão verde com o texto "ENTRAR" em branco. O fundo da tela é verde claro.

Ao carregar, é solicitado o prontuário e senha utilizada nos sistemas Desktop e Web. Para ter acesso ao aplicativo, é necessário que tenha uma conta já criada nos outros sistemas.

Em caso de não possuir a conta, entre em contato com o setor sociopedagógico do IFSP – Campus Salto e solicite uma nova conta.

SERVIÇO SOCIOPEDAGÓGICO



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Salto

FORMULÁRIO DE ENCAMINHAMENTO

Prontuário

Nome Completo

Curso Semestre/Ano

Nível ensino

Descrição do encaminhamento

ENVIAR

Ao logar com seu prontuário e senha, será aberto o formulário de encaminhamento, onde temos os seguintes campos:

- **Prontuário:**

É onde deve ser informado o prontuário do aluno que está sendo encaminhado.

- **Nome completo:**

Nome completo do aluno que está sendo encaminhado.

- **Curso:**

Curso no qual o aluno atua.

- **Semestre/Ano:**

Em que semestre e ano o aluno está cursando.

- **Nível ensino:**

Descrever se o aluno está no curso técnico noturno, médio ou superior.

- **Descrição do encaminhamento:**

São os detalhes sobre o encaminhamento do aluno. Exemplo: O motivo de o aluno estar sendo encaminhado.

Ao preencher todo o formulário, pressione o botão “Enviar” e será retornada uma mensagem informando se o envio foi concluído ou houve alguma falha.

5. Código-fonte

1. Importações padrões exigidas como alertas, caixas de textos e telas.

```
//Importações básicas do app
import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
```

2. Importações da comunicação formato JSON.

```
//importante a biblioteca para trabalhar com JSON
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

3. Importações de conexão HTTP.

```
//importante a biblioteca para trabalhar com conexão http.
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
```

4. Variáveis utilizadas.

```
//Declarando as variáveis
EditText motivo, prontuario, nome, semestre, curso, nivel;
AlertDialog alerta;
String responsavel, op;
```

5. Método que adiciona os valores informados no objeto formulário.

```
//criando uma array com os valores a serem enviados
public List<Formulario>listadados(Formulario formulario){
    List<Formulario>lista = new ArrayList<>();
    for(int i =0; i < lista.size(); i++){
        formulario.setOP(op.toString());
        formulario.setProntuario(prontuario.getText().toString());
        formulario.setMsg(motivo.getText().toString());
        formulario.setResponsavel(responsavel);
        lista.add(formulario);
    }

    return lista;
}
```

6. Método que pega as informações do aluno

```
//metodo que efetua a conexao http de busca gerando um formulario que até aqui contém somente a opção e o prontuario
public void busca(View view){
    if (prontuario.getText().toString().equals(""))
    {
        Toast.makeText(this, "há campos a serem preenchidos !", Toast.LENGTH_SHORT).show();
        return;
    }
    op = "b";
    new ConexaoHTTP().execute(gerarJSON_formulario());
}
```

7. Método utilizado na criação dos JSON.

```
//metodo que gera o Objeto e Array json.
private String gerarJSON_formulario(){
    List<Formulario>lista = listadados(new Formulario());
    JSONObject jsonObject = new JSONObject();
    JSONArray jsonArray = new JSONArray();
    try{
        for(int i = 0; i < lista.size(); i++){
            JSONObject formularioJSON = new JSONObject();
            formularioJSON.put("op", lista.get(i).getOP());
            formularioJSON.put("prontuario", lista.get(i).getProntuario());
            formularioJSON.put("motivo", lista.get(i).getMsg());
            formularioJSON.put("responsavel", lista.get(i).getResponsavel());
            jsonArray.put(formularioJSON);
        }
        jsonObject.put("formulario", jsonArray);
    }
    catch(JSONException e)
    {
        e.printStackTrace();
    }
    return jsonObject.toString();
}
```


8. Estrutura da classe criada Conexão HTTP

```
private class ConexaoHTTP extends AsyncTask<String, Void, String> {
    private ProgressDialog progress;
    @Override
    protected void onPreExecute() {
        progress = new ProgressDialog(Formulario1Activity.this);
        progress.setMessage("Enviando...");
        progress.show();
    }
    @Override
    protected String doInBackground(String... params) {
        String resposta = "";
        String dados = params[0];
        InputStream input = null;
        HttpURLConnection conexao = null;
        StringBuilder resultado = null;
        try {
            //URL urlCon = new URL("http://sgaphp.site88.net/SGA/webservice.php");
            URL urlCon = new URL("http://localhost/websevice.php");
            conexao = (HttpURLConnection) urlCon.openConnection();
            conexao.setReadTimeout(10000);
            conexao.setConnectTimeout(15000);
            conexao.setRequestMethod("POST");
            conexao.setDoInput(true);
            conexao.setDoOutput(true);
            conexao.setRequestProperty("Content-Type", "application/json");
            conexao.setRequestProperty("Accept", "application/json");

            conexao.connect();

            BufferedWriter output = new BufferedWriter(new OutputStreamWriter(conexao.getOutputStream()));
            output.write(dados);
            output.close();
        }
    }
}
```

9. Método de validações antes de enviar.

```
//metodo que sincroniza o enviar com o progress bar de encaminhamento de formulario para webservice.
public void sincronizar(View view){
    if {
        prontuario.getText().toString().equals("") ||
        motivo.getText().toString().equals("") ||
        semestre.getText().toString().equals("") ||
        nivel.getText().toString().equals("") ||
        nome.getText().toString().equals("")
    }
    {
        op = "i";
        Toast.makeText(this, "há campos a serem preenchidos !", Toast.LENGTH_SHORT).show();
        return;
    }
    else {
        try {
            AlertDialog alertDialog = sincronizarDados();
            alertDialog.show();
        } catch (Exception e) {
            Toast.makeText(this, "Erro no meio do processo! " + e.toString(), Toast.LENGTH_LONG).show();
        }
    }
}
```

10. Método que executa o envio do relatório final.

```
//metodo que executa a conexao http para enviar o formulario final
private void enviarDados(){
    new ConexaoHTTP().execute(gerarJSON_formulario());
}
```