# Guia para desenvolvedores – Instalação do sistema de Gestão de Atividades Simultâneas (GAS)

Atualizado: 12/07/2023

Renan F Maziero

renanfmaziero@gmail.com

## Sumário

# Objetivo

Esse guia rápido serve para orientar quem for manter o sistema GAS futuramente. Consideramos que uma máquina foi formatada e o sistema GAS precisa ser instalado do zero. Tudo o que temos nesse momento é somente um sistema operacional Linux rodando. O sistema GAS roda em ***Spring Boot (backend)*** e ***Angular (frontend)***. Basicamente precisaremos instalar e configurar:

- ⚠ Nginx (>=1.18.0)
- ⚠ Java 11 (tem que ser a 11.0.2)
- ⚠ Maven (>=3.9.3)
- ⚠ Git (>=2.34.1)
- ⚠ Docker (>=24.0.2)
- ⚠ Node.js (>=18.16.1)
- ⚠ NPM (>=9.5.1)
- ⚠ Angular CLI (>=16.1.1)

# Configurações iniciais (remoto)

Siga os passos a seguir para instalação das dependências do GAS no servidor remoto:

1. Solicite acesso SSH ao departamento de informática da FT para acessar a máquina. https://www.ft.unicamp.br/pt-br/secoes/informática

Eles vão fornecer os dados de acesso. No presente momento os dados são:

```
usuário: <usuário_fornecido>
senha: <senha_fornecida>
servidor: geicon.ft.unicamp.br (143.106.243.175)
porta ssh: 22000
```

Por segurança, só é permito acessar a máquina de dentro da rede VPN da Unicamp. Considere ler como configurar a VPN no link:
https://www.ccuec.unicamp.br/ccuec/servicos/acesso_remoto_vpn

2. Uma vez com o acesso garantido via SSH e logado na máquina recém formatada, considere fazer as configurações iniciais de segurança da máquina: https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04
3. Ajuste a data e hora: *"sudo timedatectl set-timezone America/Porto_Velho"*
4. Libere as portas no Firewall:

```
sudo ufw allow 22000

sudo ufw allow 8080

sudo ufw allow 8081

sudo ufw allow 80

sudo ufw allow 4200

sudo ufw allow 443

sudo ufw allow 587

sudo ufw allow 5432
```

Sendo a porta 22000 (SSH), 8080 (Docker), 8081 (API Java Spring), 80 (Nginx), 4200 (Angular), 443 (SSL), 587 (Disparo de e-mails via backend) e 5432 (Banco de dados).

⚠ Importante! Além de liberar com os comandos acima, será necessário solicitar à seção de informática da FT que também liberem o tráfego nessas portas por lá.

# Instalação backend (remoto)

## Nginx

5. Instalar o servidor Nginx: https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-22-04

⚠ Importante! Não se esqueça de, na parte de configurar o server blocks do domínio, servir sempre o arquivo *"index.html"*, senão as rotas do Angular retornaram 404.

```
sudo nano /etc/nginx/sites-available/geicon.ft.unicamp.br
(...)
location / {
  try_files $uri $uri/ /index.html =404;
}
```

## Java 11

6. Instalar o Java 11: https://www.digitalocean.com/community/tutorials/install-maven-linux-ubuntu#installing-jdk-on-linux-ubuntu

   ⚠ Importante! Na hora da instalação, o link do binário do Java deve ser da versão 11.0.2.

   ⚠ Importante! Não se esqueça de definir as variáveis de ambiente *"JAVA_HOME"* e *"PATH"* depois da instalação. Como nos códigos abaixo:

```
wget https://download.java.net/java/GA/jdk11/9/GPL/openjdk-11.0.2_linux-x64_bin.tar.gztar -xvf openjdk-11.0.2_linux-x64_bin.tar.gz

mv jdk-11.0.2 /opt/
JAVA_HOME='/opt/jdk-11.0.2'
PATH="$JAVA_HOME/bin:$PATH"
export PATH
java –version
```

## Maven

7. Instalar o Maven: https://www.digitalocean.com/community/tutorials/install-maven-linux-ubuntu#installing-maven-on-linux-ubuntu

   ⚠ Importante! Não se esqueça de definir as variáveis *"M2_HOME"* e *"PATH"* do Maven descritas no tutorial acima.

### Git

8. Instalar e configurar o GIt: https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu-20-04

### Docker

9. Instalar o Docker: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-22-04

## Instalação frontend (remoto)

### Node.js + NPM

10. Instalar o Node.js + NPM: https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-20-04#option-1-installing-node-js-with-apt-from-the-default-repositories

    ⚠ Importante! A versão do Node.js precisar ser *"**>=18.x.x**"*. Então, lembre-se de alterar na hora da instalação!

### Angular

11. Instalar o Angular CLI: https://angular.io/cli. Basicamente é só rodar o comando:

```
npm install -g @angular/cli
```

## Rodando localmente (testes)

Consideramos um ambiente Windows para desenvolvimento local.

12. Crie os diretórios locais para os projetos. Por exemplo:

    *"C:/gas/frontend/"*

    *"C:/gas/backend/"*

13. Clone os projetos do Git para dentro dos seus respectivos diretórios locais:

```
cd C:/gas/frontend/
git clone https://github.com/renanmaziero/gestao-extensao-frontend.git

cd C:/gas/backend/
git clone https://github.com/renanmaziero/gestao-extensao-backend.git
```

## Rodando backend local

14. Instalar o JDK 11.0.2: https://jdk.java.net/archive/ ou use o link direto https://download.java.net/java/GA/jdk11/9/GPL/openjdk-11.0.2_windows-x64_bin.zip

15. Instalar o Docker: https://docs.docker.com/desktop/install/windows-install/
16. Inicie o Docker

Ao terminar de instalar o JDK 11.0.2 e iniciar o Docker, abra o projeto pelo IntelliJ. Crie um arquivo nomeado ***"application-local.properties"*** e vá em ***"Edit Configurations"***, em seguida ***"Environment variables"*** e coloque ***"PROFILE=local"***. No arquivo criado, você deverá colocar as seguintes propriedades de configuração para estar apto a rodar o projeto localmente:

```
spring.profiles.active=${PROFILE}

server.port=8081

spring.datasource.url=jdbc:postgresql://localhost:5432/sag-extensao

spring.datasource.username=sag

spring.datasource.password=develop

spring.jpa.show-sql=true

spring.jpa.generate-ddl=true

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true

# Hibernate ddl auto (create, create-drop, validate, update)

spring.jpa.hibernate.ddl-auto=update

spring.servlet.multipart.max-file-size=2MB
```

```
spring.servlet.multipart.max-request-size=2MB

gestao.extensao.jwtSecret=ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEApi6dplhPmrm/nR/2UvPWYAwxY7JCC+KL3zN1qIb2s+6V284
xfGcE3BACae6NdrrcZX8rpTwdZOQKcQdCrWb+GkoA2LNNFEEkj9h3SCvPSVO8PEwlu5wJbRPyfVK
ZIvuei1hD6y0QQbIUtwA7aPTUTMaMqSL85v8Zp/GlT3ZvKu3sqQhlL3KMpAF928EzGy2wGTpdGoE
Ln1W0VNMamUM+QhBJz2CZf7tDebi2h2eRr1HKoDciN+WctB5o7mcMZL6snOuVZgax0d7LdwAR
X0a4tPjpos4ZH3KFbJTkkKKMubJwMijbd5y/6sMnrAWGn2orWEctX80tsU6CcOzR30OC1Q== rsa-
key-20210726
gestao.extensao.tempoExpiracaoJwt=3600000000
spring.mail.host=smtp.mailgun.org
spring.mail.port=587
spring.mail.username=postmaster@mail.unicamp-extensao.com.br
spring.mail.from.username=comissao.extensao.ft@gmail.com

spring.mail.password=65d4e19f4f8e310d4c271eda267c4696-1d8af1f4-aa5bb126

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

spring.mail.smtp.socketFactory.port=587

spring.mail.smtp.socketFactory.fallback=true

spring.mail.smtp.starttls.enable=true

spring.mail.smtp.starttls.required=true

spring.mail.smtp.ssl.enable=false

spring.mail.domain=mail.unicamp-extensao.com.br

spring.mail.api.key=15d575660e5f59c30d02b465c285b53f-e49cc42c-8b1635ec

coordenador.nome=GERUSA DE CÁSSIA SALADO

frontend.url=http://localhost:4200/
```

⚠ Importante! Os dados do *"application-local.properties"* estão expostos aqui somente por estarem localmente. Jamais exponha-os em produção. Em produção, declare variáveis de ambiente para esconder os dados.

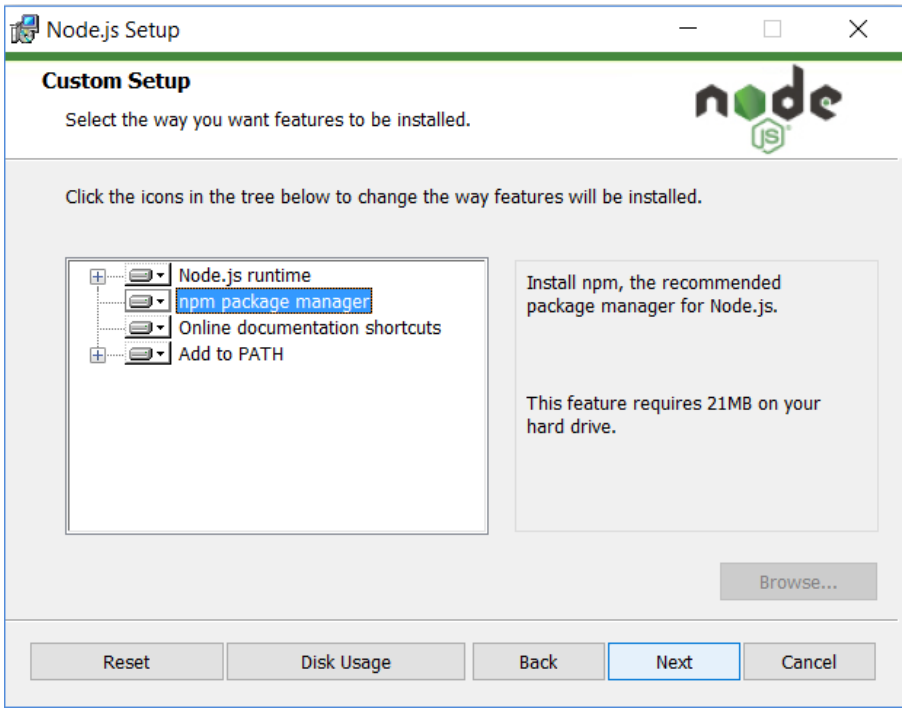
17. Execute pelo IntelliJ com *"Shift + F10"*. Nesse momento o IntelliJ irá automaticamente* usar o Maven para compilar e subir o Spring, além de subir o container no Docker com o banco de dados. Se tudo estiver certo, a API já estará rodando na porta 8081 em seu computador.

⚠ Importante! Nesse momento o projeto será compilado usando o arquivo de configuração *"src/main/resources/application-local.properties"*.

⚠ *dependendo de como foi instalado o Intellij. Caso ocorra algum erro, rode manualmente os comandos *"mvn clean install"* e *"docker compose up"* na raiz do projeto *"C:/gas/backend/gestao-extensao-backend/"*.

## Rodando frontend local

18. Instalar o Node.js + NPM: https://nodejs.org/en/download
    No instalador para Windows, o Node.js e o NPM são instalados simultaneamente. Não se esqueça de habilitar o NPM na instalação como a imagem abaixo:




⚠ Importante! Confirme as instalações com os comandos *"node –v"* e *"npm –v"* no terminal. Se tudo estiver certo, exibirá as respectivas versões.

19. Instalar o Angular CLI: https://angular.io/guide/setup-local. Basicamente é só rodar os comandos no Windows PowerShell:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
npm install -g @angular/cli
npm install
```

Para compilar e subir o frontend local do Angular, rode os comandos:

20. Build local para testes (somente uma vez, pois a cada alteração no projeto local, o Angular recompilará automaticamente, vigorando as alterações):

```
cd C:/gas/frontend/
$env:NODE_OPTIONS="--openssl-legacy-provider"
ng serve --host 0.0.0.0 --disable-host-check
```

⚠ Importante! Nesse momento o projeto subirá em *"localhost:4200"* usando o arquivo de configuração *"src\environments\environment.ts"*.

⚠ Este é um servidor simples para uso em testes ou depuração de aplicativos Angular localmente. Não foi revisado por problemas de segurança. Não deve ser feito em produção. *"Ctrl + C"* encerra o servidor.

## Rodando remotamente (produção)

21. Crie os diretórios para os projetos. Por exemplo:
    *"mkdir -p /gas/frontend/"*
    *"mkdir -p /gas/backend/"*

22. Clone os projetos do Git para dentro dos seus respectivos diretórios:

```
cd /gas/frontend/
git clone https://github.com/renanmaziero/gestao-extensao-frontend.git
```

```
cd /gas/backend/
git clone https://github.com/renanmaziero/gestao-extensao-backend.git
```

## Rodando backend remoto

O sistema usa a ferramenta de automação de compilação Maven, que se encarregará de tudo. Primeiramente, será necessário compilar e depois executar a API. Rode os comandos:

23. Para compilar (sempre que novas alterações no projeto Java Spring forem feitas. Senão não entraram em vigor):

```
cd /gas/backend/gestao-extensao-backend/
mvn clean install
```

⚠ Importante! Nesse momento o arquivo "***.jar***" será gerado e armazenado na pasta "***/target***", usando o arquivo de configuração "***src/main/resources/application.properties***".

⚠ Por motivo de segurança, as credenciais no arquivo ***"application.properties"*** estão escondidas em variáveis de ambiente. Veja ao final desse documento como defini-las.

24. Antes de executar, é necessário subir o banco de dados. Para iniciar o container Docker com o banco de dados (necessário apenas na primeira vez, mesmo que a máquina reinicie, já que o serviço ficará rodando sempre):

```
cd /gas/backend/gestao-extensao-backend/
docker compose up
```

25. Para executar o ***".jar"*** e subir a API (sempre que novas alterações no projeto Java Spring forem feitas. Senão não entraram em vigor):

```
nohup java -jar /gas/backend/gestao-extensao-backend/target/tcc-0.0.1-SNAPSHOT.jar
> /gas/backend/gestao-extensao-backend/target/log-spring.txt &
```

⚠ Importante! ***"nohup"*** permite manter a API rodando mesmo que você feche o terminal. Sem isso vai derrubar a API quando fechar a sessão.

⚠ Importante! Todo o log da execução da API será armazenado no arquivo ***"log-spring.txt"***

⚠ Importante! O Símbolo de "***&"*** envia o processo para segundo plano. Isso permite que você continue usando a sessão. Anote o número do processo ou encontre-o no ***"log-spring.txt"*** ao lado da coluna ***"INFO"***. Caso, por algum motivo, precise encerrá-lo, rode o comando "***kill <numero_processo>",*** ou traga o processo para o primeiro plano com o comando ***"fg"*** e depois encerre com ***"Ctrl + C"***.

⚠ Importante! Se por algum motivo precisar reiniciar a máquina, lembre-se de subir a API novamente seguindo o passo 25 acima.

⚠ Importante! Os comandos só funcionaram se as variáveis de ambiente foram definidas. Veja os passos 6 e 7.

## Rodando frontend remoto

O sistema usa a ferramenta de automação de compilação e build do próprio Angular, que se encarregará de tudo. Rode os comandos:

26. Para build em produção (sempre que novas alterações no projeto Angular forem feitas. Senão não entraram em vigor):

```
export NODE_OPTIONS=--openssl-legacy-provider
ng b --configuration=production --output-path /var/www/geicon.ft.unicamp.br/html/
```

⚠ Importante! Nesse momento os arquivos estáticos serão gerados e hospedados no servidor Nginx, usando o arquivo de configuração "***src\environments\environment.prod.ts"***.

# Considerações finais

27. Considere armazenar as variáveis de ambiente no Linux, para não ter que definir toda vez antes das execuções. Abra o arquivo "***.bashrc"*** e adicione ao final do arquivo e salve:

```
sudo nano ~/.bashrc
export JAVA_HOME='/opt/jdk-11.0.2'
export PATH="$JAVA_HOME/bin:$PATH"
export M2_HOME='/opt/apache-maven-3.9.3'
export PATH="$M2_HOME/bin:$PATH"
export NODE_OPTIONS=--openssl-legacy-provider
```

⚠ Importante! Para refletir as alterações no bash rode o comando ***"source .bashrc"***

⚠ Importante! Para visualizar o valor da variável rode o comando ***"echo ${NOME_VAR}"***

28. Será necessário adicionar dois registros na tabela ***"profile"*** do banco de dados. Adicione conforme abaixo:

| id | name |
|---|---|
| 1 | ROLE_ADMIN |
| 2 | ROLE_USER |

⚠ Importante! Devem ser adicionados esses registros tanto no banco de dados local para testes, quanto no remoto em produção. Sem esses registros não funcionará a tela de cadastro de novos usuários.

29. Será necessário adicionar um registro na tabela ***"parametrizacao"*** do banco de dados. Adicione conforme abaixo:

| id | max_hr_mensais_convenio | max_hr_ministradas_curso | max_hr_semanais_convenio | max_hr_semestrais_convenio | max_hr_semestrais_curso | max_hr_semestrais_regencia |
|---|---|---|---|---|---|---|
| 1 | 48 | 240 | 12 | 60 | 60 | 60 |

⚠ Importante! Deve ser adicionado esse registro tanto no banco de dados local para testes, quanto no remoto em produção. Sem esse registro não funcionará a tela de cadastro de novos usuários.

30. Para rodar comandos no banco de dados, use seu programa preferido e, forneça os dados de acesso:

    Local

```
Host: localhost
Porta: 5432
Usuário: sag
Senha: develop
Banco: postgres
```

    Remoto

```
Host: geicon.ft.unicamp.br
Porta: 5432
Usuário: sag
Senha: develop
Banco: postgres
```

⚠ Esses dados de acesso são definidos no arquivo *"docker-compose.yml"*.

⚠ Se preferir, use a URL (local): *"jdbc:postgresql://localhost:5432/postgres"*

⚠ Se preferir, use a URL (remoto): *"jdbc:postgresql://geicon.ft.unicamp.br:5432/postgres"*

31. Para definir as variáveis de ambiente que serão usadas no arquivo *"application.properties",* abra o arquivo "*.bashrc"* e adicione ao final do arquivo e salve:

```
sudo nano ~/.bashrc
export SPRING_PROFILES_ACTIVE=prod
```

```
export SERVER_PORT=8081
export SPRING_DATASOURCE_URL=jdbc:postgresql://geicon.ft.unicamp.br:5432/postgres
export SPRING_DATASOURCE_USERNAME=sag
export SPRING_DATASOURCE_PASSWORD=develop
export SPRING_JPA_GENERATE_DDL=true
export SPRING_JPA_PROPERTIES_HIBERNATE_JDBC_LOB_NON_CONTEXTUAL_CREATION=true
export SPRING_JPA_HIBERNATE_DDL_AUTO=update
export SPRING_SERVLET_MULTIPART_MAX_FILE_SIZE=2MB
export SPRING_SERVLET_MULTIPART_MAX_REQUEST_SIZE=2MB
export GESTAO_EXTENSAO_JWTSECRET "ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEApi6dplhPmrm/nR/2UvPWYAwxY7JCC+KL3zN1qIb2s+6V284
xfGcE3BACae6NdrrcZX8rpTwdZOQKcQdCrWb+GkoA2LNNFEEkj9h3SCvPSVO8PEwlu5wJbRPyfVK
ZIvuei1hD6y0QQbIUtwA7aPTUTMaMqSL85v8Zp/GlT3ZvKu3sqQhlL3KMpAF928EzGy2wGTpdGoE
Ln1W0VNMamUM+QhBJz2CZf7tDebi2h2eRr1HKoDciN+WctB5o7mcMZL6snOuVZgax0d7LdwAR
X0a4tPjpos4ZH3KFbJTkkKKMubJwMijbd5y/6sMnrAWGn2orWEctX80tsU6CcOzR30OC1Q== rsa-
key-20210726"
export GESTAO_EXTENSAO_TEMPOEXPIRACAOJWT=3600000000
export SPRING_MAIL_HOST=smtp.mailgun.org
export SPRING_MAIL_PORT=587
export SPRING_MAIL_USERNAME=postmaster@mail.unicamp-extensao.com.br
export SPRING_MAIL_FROM_USERNAME=comissao.extensao.ft@gmail.com
export SPRING_MAIL_PASSWORD=65d4e19f4f8e310d4c271eda267c4696-1d8af1f4-aa5bb126
export SPRING_MAIL_PROPERTIES_MAIL_SMTP_AUTH=true
export SPRING_MAIL_PROPERTIES_MAIL_SMTP_STARTTLS_ENABLE=true
export SPRING_MAIL_SMTP_SOCKETFACTORY_PORT=587
export SPRING_MAIL_SMTP_SOCKETFACTORY_FALLBACK=true
export SPRING_MAIL_SMTP_STARTTLS_ENABLE=true
export SPRING_MAIL_SMTP_STARTTLS_REQUIRED=true
export SPRING_MAIL_SMTP_SSL_ENABLE=false
export SPRING_MAIL_DOMAIN=mail.unicamp-extensao.com.br
```

```
export SPRING_MAIL_API_KEY=15d575660e5f59c30d02b465c285b53f-e49cc42c-8b1635ec

export COORDENADOR_NOME="Gerusa De Cássia Salado"

export FRONTEND_URL=http://geicon.ft.unicamp.br/
```

- ⚠ Importante! Para refletir as alterações no bash rode o comando *"source .bashrc"*
- ⚠ Importante! Para visualizar o valor da variável rode o comando *"echo ${NOME_VAR}"*