**Finalizando os controlers da API:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using Projeto.Application.ViewModels;

using Projeto.Application.Contracts;

namespace Projeto.Presentation.Controllers

{

[RoutePrefix("api/Contato")] //ENDPOINT

public class ContatoController : ApiController

{

//atributo

private readonly IContatoAppService appService;

//construtor para injeção de dependência

public ContatoController(IContatoAppService appService)

{

this.appService = appService;

}

[HttpPost]

public HttpResponseMessage Post(ContatoCadastroViewModel model)

{

if (ModelState.IsValid)

{

try

{

appService.Cadastrar(model);

return Request.CreateResponse

(HttpStatusCode.OK, "Contato cadastrado com sucesso.");

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

else

{

return Request.CreateResponse(HttpStatusCode.BadRequest);

}

}

[HttpPut]

public HttpResponseMessage Put(ContatoEdicaoViewModel model)

{

if (ModelState.IsValid)

{

try

{

appService.Atualizar(model);

return Request.CreateResponse

(HttpStatusCode.OK, "Contato atualizado com sucesso.");

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

else

{

return Request.CreateResponse(HttpStatusCode.BadRequest);

}

}

[HttpDelete]

public HttpResponseMessage Delete(int id)

{

try

{

appService.Excluir(id);

return Request.CreateResponse

(HttpStatusCode.OK, "Contato excluído com sucesso.");

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

[HttpGet]

public HttpResponseMessage GetAll()

{

try

{

var model = appService.ConsultarTodos();

return Request.CreateResponse(HttpStatusCode.OK, model);

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

[HttpGet]

public HttpResponseMessage GetById(int id)

{

try

{

var model = appService.ConsultarPorId(id);

return Request.CreateResponse(HttpStatusCode.OK, model);

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

}

}

using Projeto.Application.Contracts;

using Projeto.Application.ViewModels;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace Projeto.Presentation.Controllers

{

[RoutePrefix("api/Compromisso")]

public class CompromissoController : ApiController

{

//atributo

private readonly ICompromissoAppService appService;

//construtor para injeção de dependência

public CompromissoController(ICompromissoAppService appService)

{

this.appService = appService;

}

[HttpPost]

public HttpResponseMessage Post(CompromissoCadastroViewModel model)

{

if (ModelState.IsValid)

{

try

{

appService.Cadastrar(model);

return Request.CreateResponse

(HttpStatusCode.OK, "Compromisso cadastrado

com sucesso.");

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

else

{

return Request.CreateResponse(HttpStatusCode.BadRequest);

}

}

[HttpPut]

public HttpResponseMessage Put(CompromissoEdicaoViewModel model)

{

if (ModelState.IsValid)

{

try

{

appService.Atualizar(model);

return Request.CreateResponse

(HttpStatusCode.OK, "Compromisso atualizado

com sucesso.");

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

else

{

return Request.CreateResponse(HttpStatusCode.BadRequest);

}

}

[HttpDelete]

public HttpResponseMessage Delete(int id)

{

try

{

appService.Excluir(id);

return Request.CreateResponse

(HttpStatusCode.OK, "Compromisso excluído com sucesso.");

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

[HttpGet]

public HttpResponseMessage GetAll()

{

try

{

var model = appService.ConsultarTodos();

return Request.CreateResponse(HttpStatusCode.OK, model);

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

[HttpGet]

public HttpResponseMessage GetById(int id)

{

try

{

var model = appService.ConsultarPorId(id);

return Request.CreateResponse(HttpStatusCode.OK, model);

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

}

}

using Projeto.Application.Contracts;

using Projeto.Application.ViewModels;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace Projeto.Presentation.Controllers

{

[RoutePrefix("api/Endereco")]

public class EnderecoController : ApiController

{

//atributo

private readonly IEnderecoAppService appService;

//construtor para injeção de dependência

public EnderecoController(IEnderecoAppService appService)

{

this.appService = appService;

}

[HttpPost]

public HttpResponseMessage Post(EnderecoCadastroViewModel model)

{

if (ModelState.IsValid)

{

try

{

appService.Cadastrar(model);

return Request.CreateResponse

(HttpStatusCode.OK, "Endereço cadastrado com sucesso.");

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

else

{

return Request.CreateResponse(HttpStatusCode.BadRequest);

}

}

[HttpPut]

public HttpResponseMessage Put(EnderecoEdicaoViewModel model)

{

if (ModelState.IsValid)

{

try

{

appService.Atualizar(model);

return Request.CreateResponse

(HttpStatusCode.OK, "Endereço atualizado com sucesso.");

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

else

{

return Request.CreateResponse(HttpStatusCode.BadRequest);

}

}

[HttpDelete]

public HttpResponseMessage Delete(int id)

{

try

{

appService.Excluir(id);

return Request.CreateResponse

(HttpStatusCode.OK, "Endereço excluído com sucesso.");

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

[HttpGet]

public HttpResponseMessage GetAll()

{

try

{

var model = appService.ConsultarTodos();

return Request.CreateResponse(HttpStatusCode.OK, model);

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

[HttpGet]

public HttpResponseMessage GetById(int id)

{

try

{

var model = appService.ConsultarPorId(id);

return Request.CreateResponse(HttpStatusCode.OK, model);

}

catch (Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

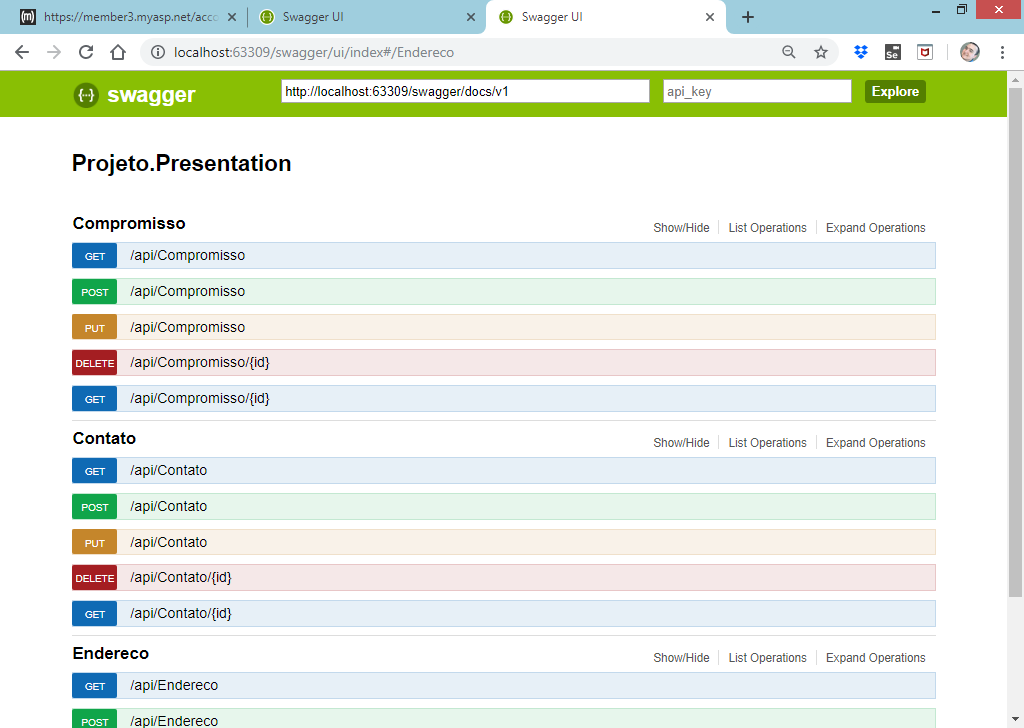
}

}

}

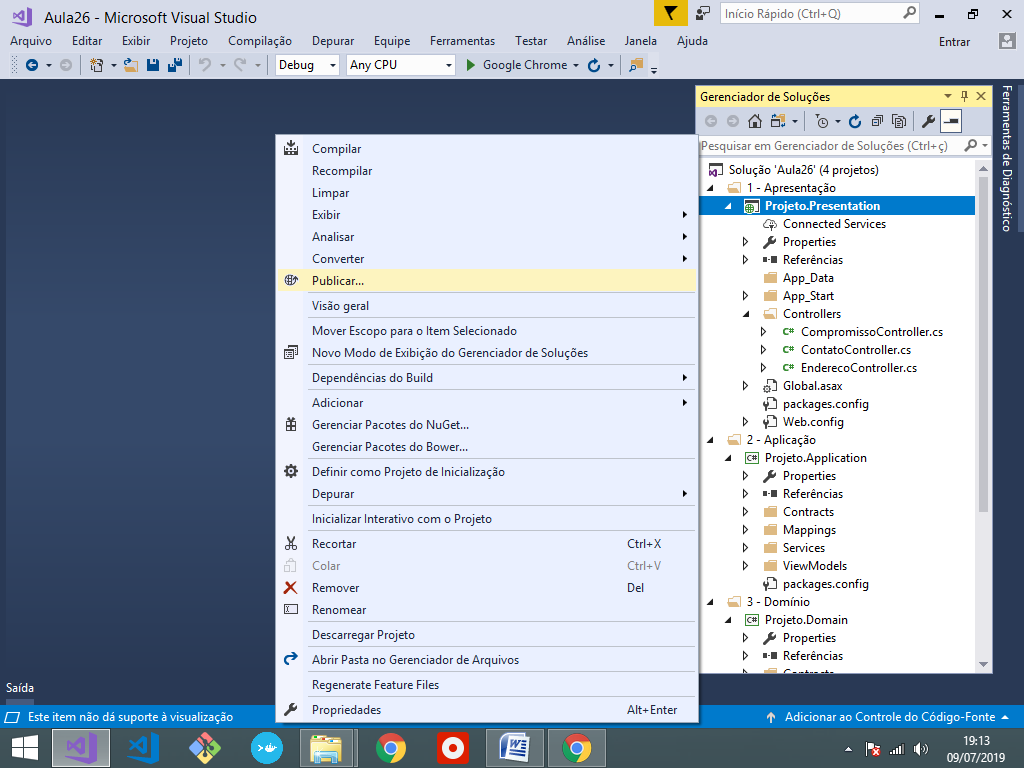
**Testando a execução da API:**

<http://localhost:63309/swagger/ui/index>

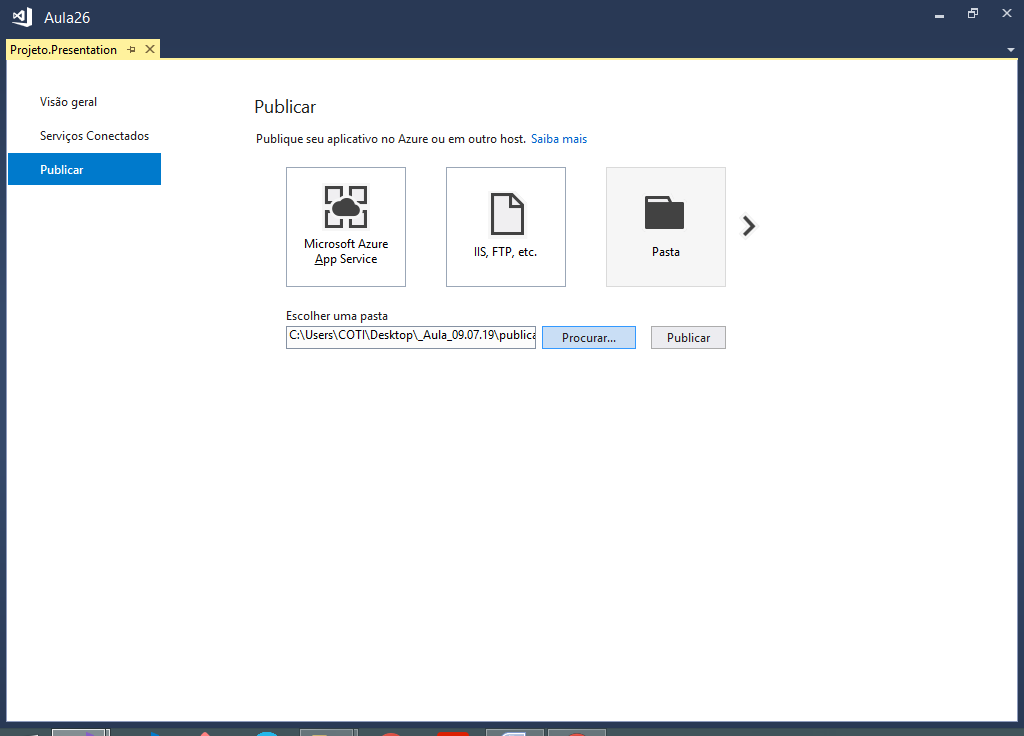


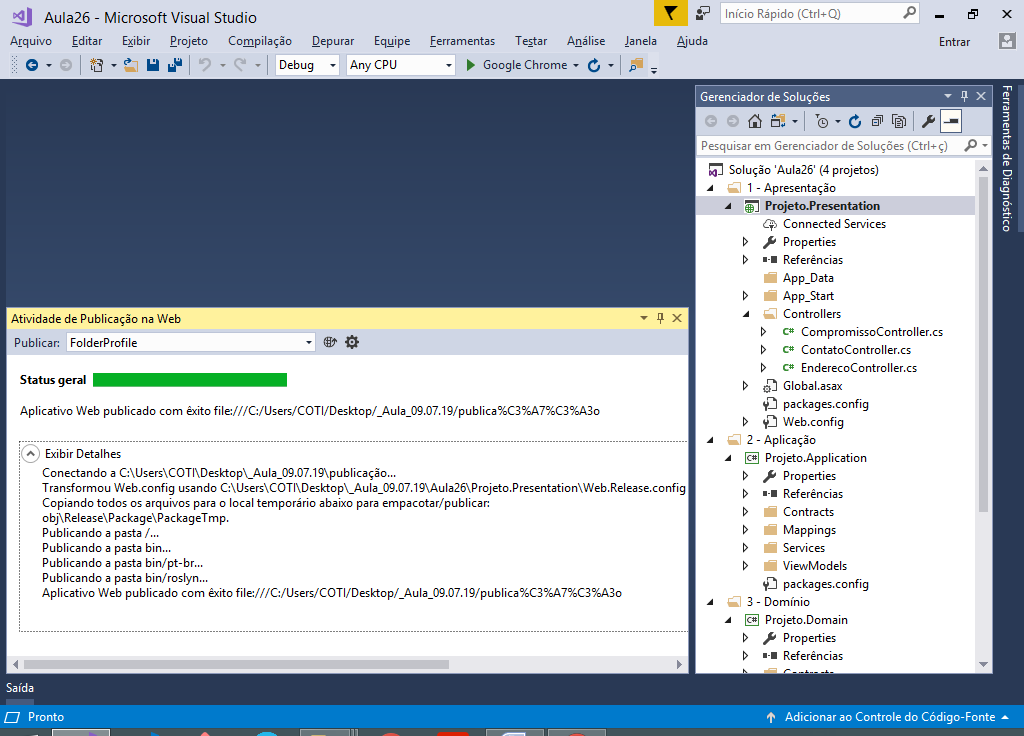
**Publicando o projeto:**

Exportando os arquivos de publicação do projeto para uma pasta local

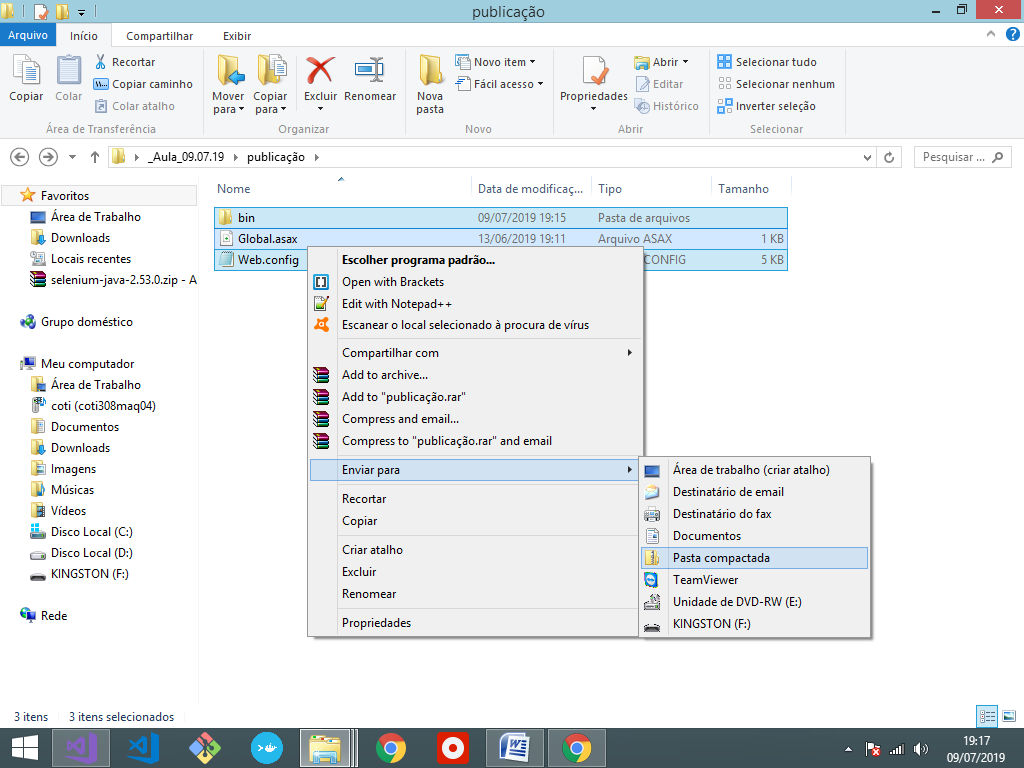


Selecione "**Pasta**" e escolha o diretório para o qual deseja exportar os arquivos de publicação:





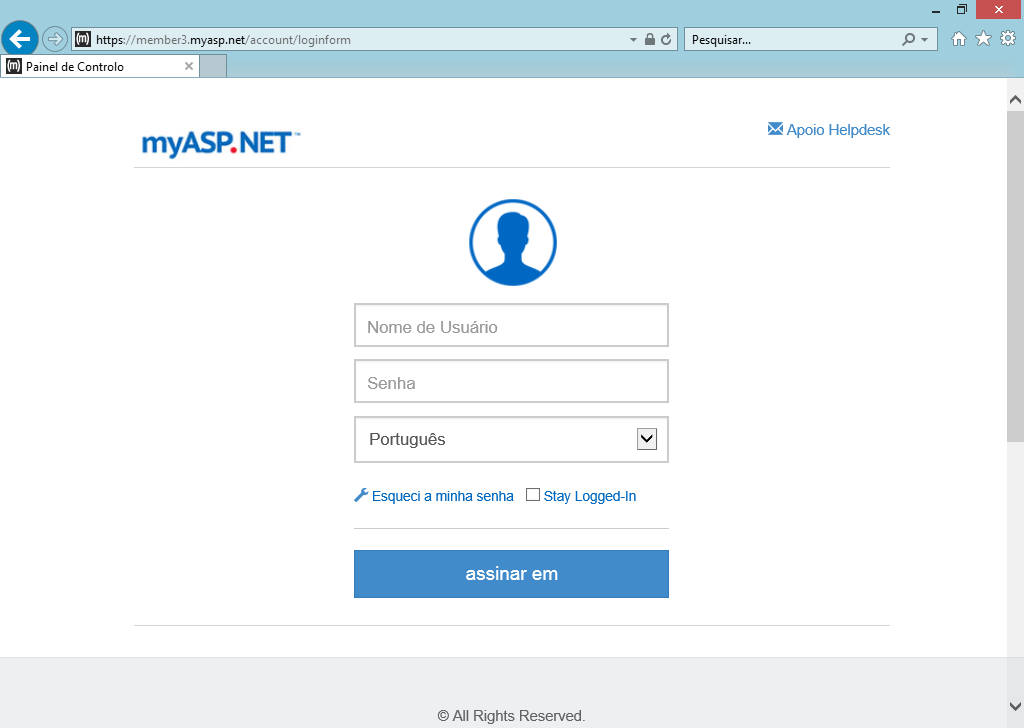
**Zipando os arquivos gerados:**



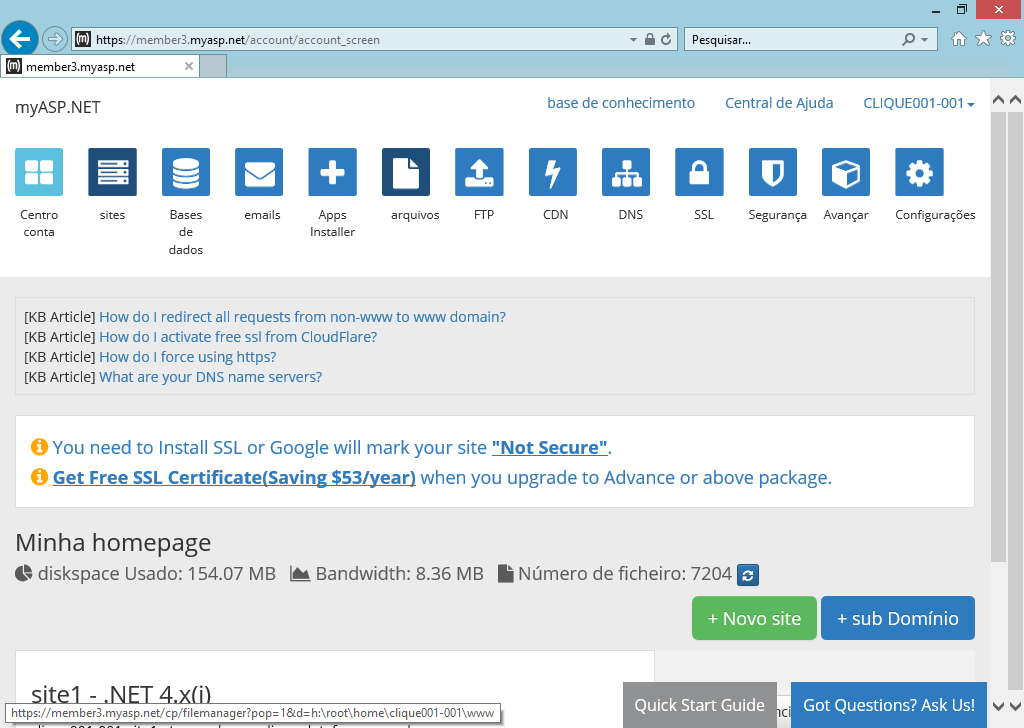
https://www.myasp.net/



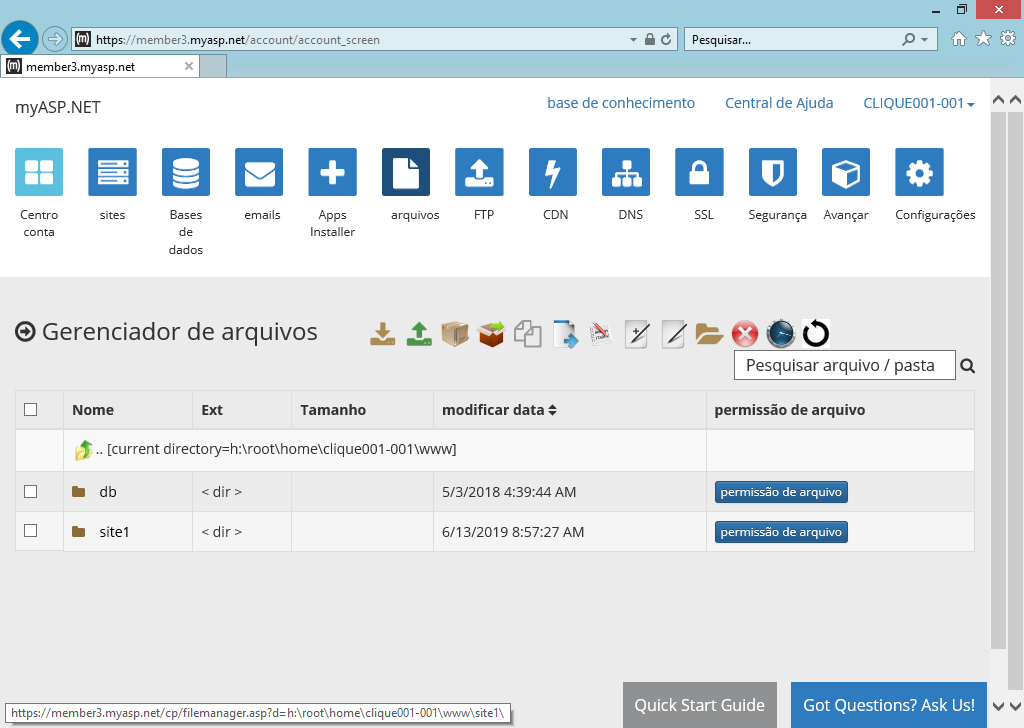
**Acessando o serviço de hospedagem:**



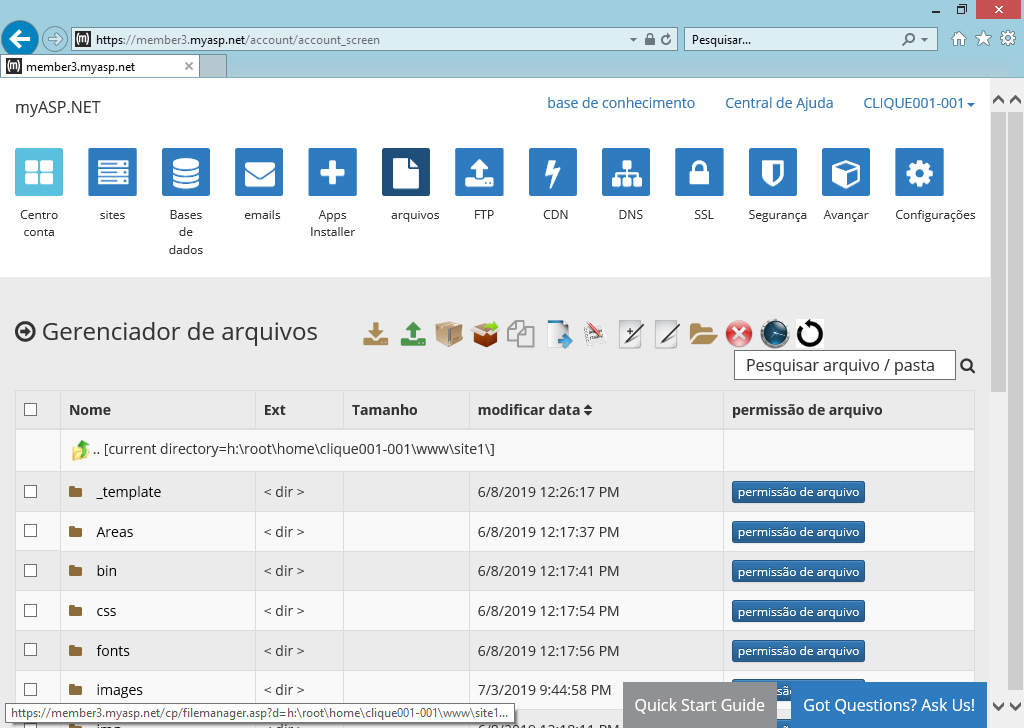
Acesse a aba "**Arquivos**"



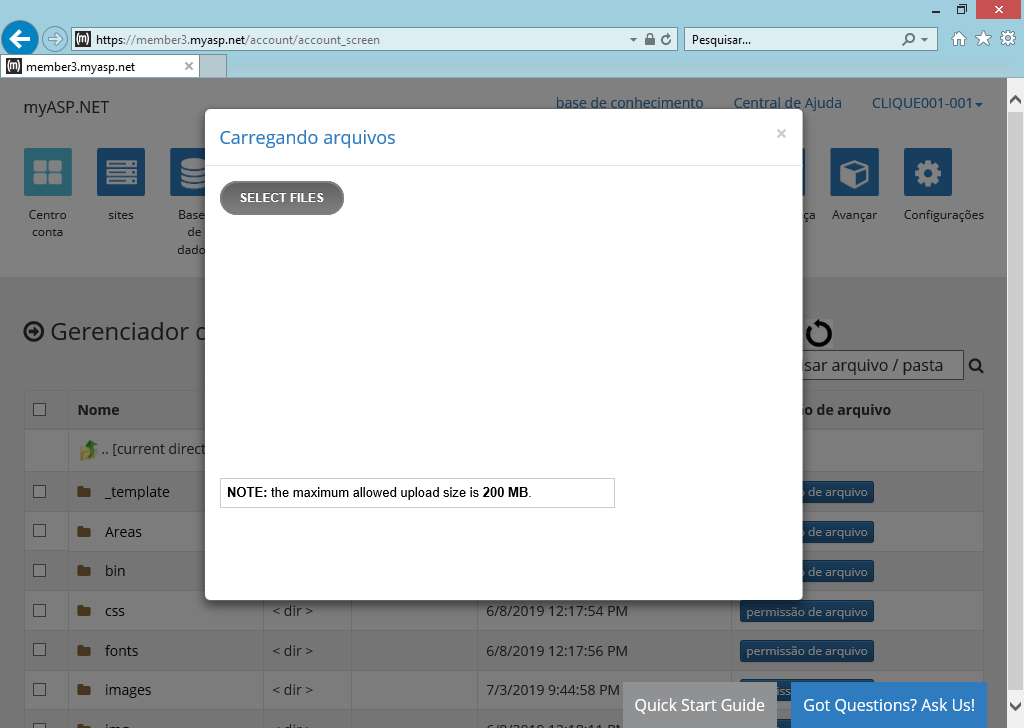
Acesse a pasta: **site1**



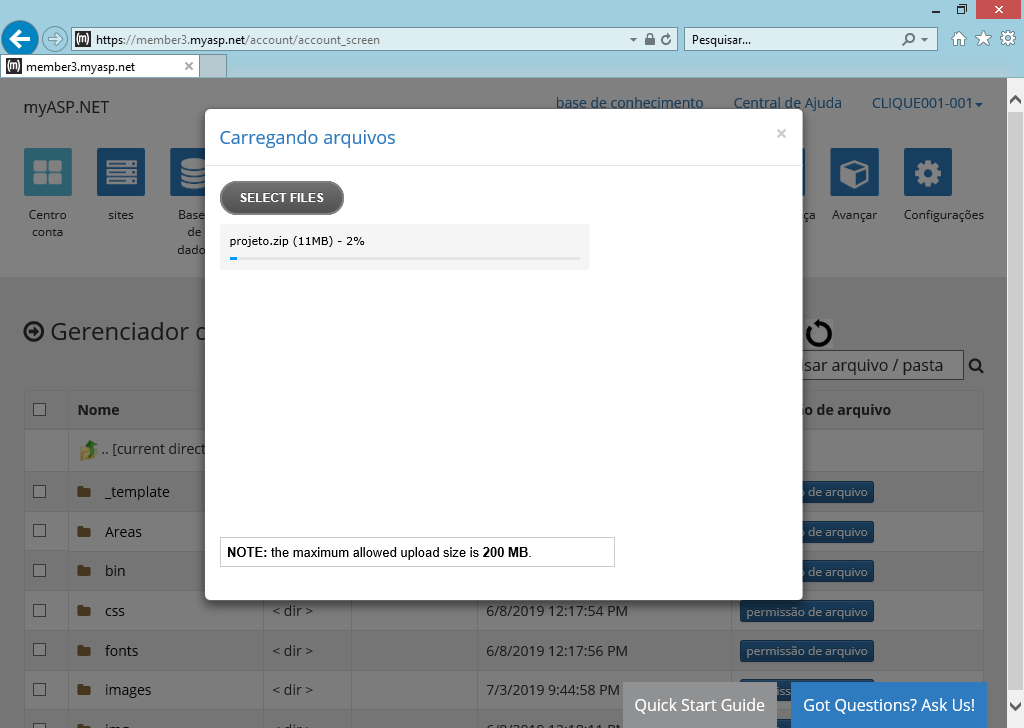
Escolha a opção "**Upload**"



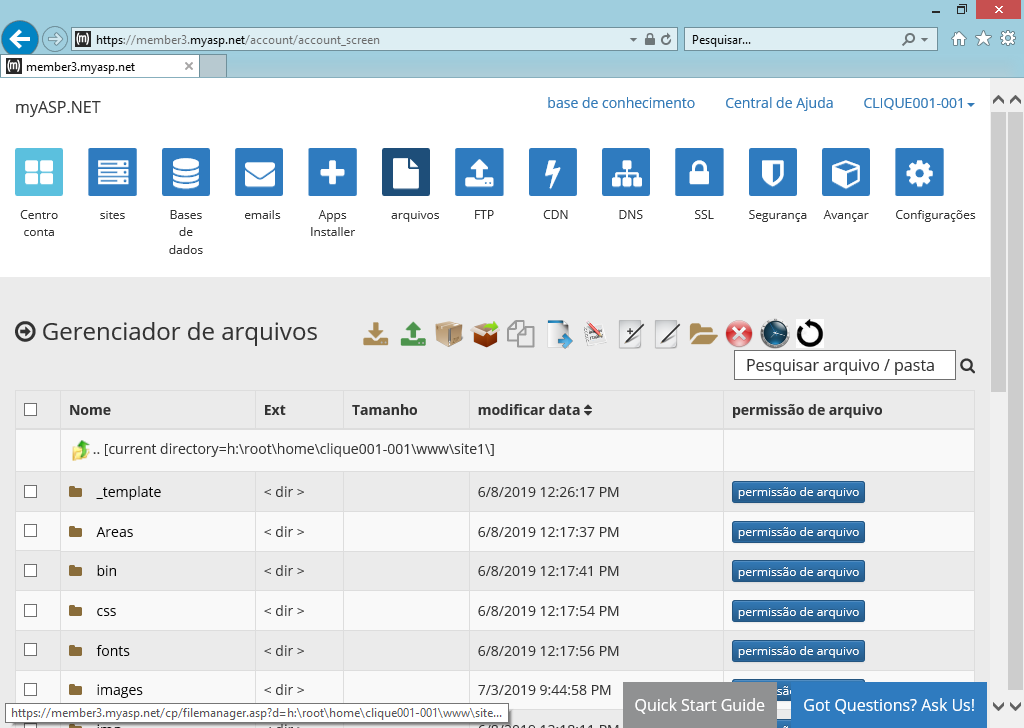
Clique em "**Selecionar Arquivos**"



Selecione envie o arquivo .zip´



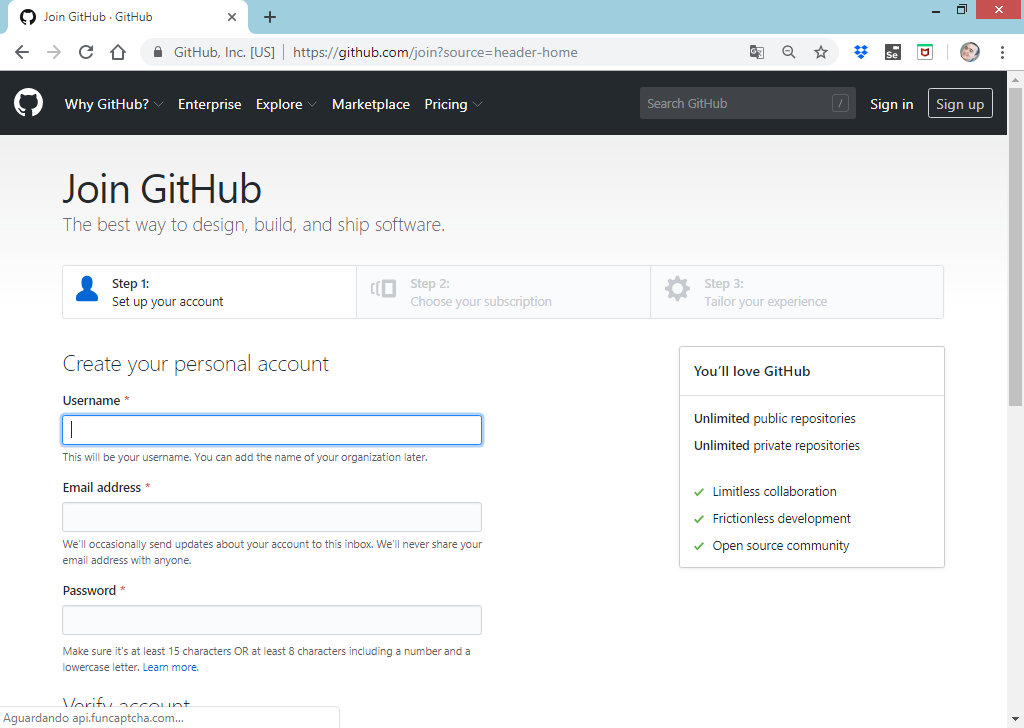
**Descompacte o arquivo zip:**



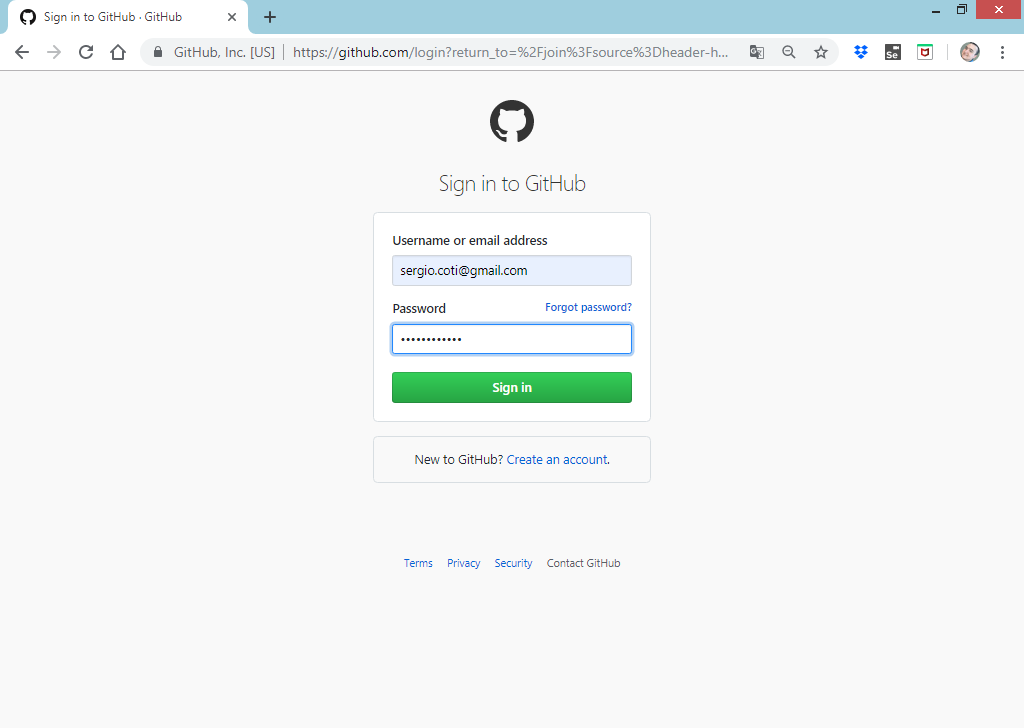
Criando uma conta no GitHub (<https://github.com/>)

Repositório de código (Portfolio de programadores)

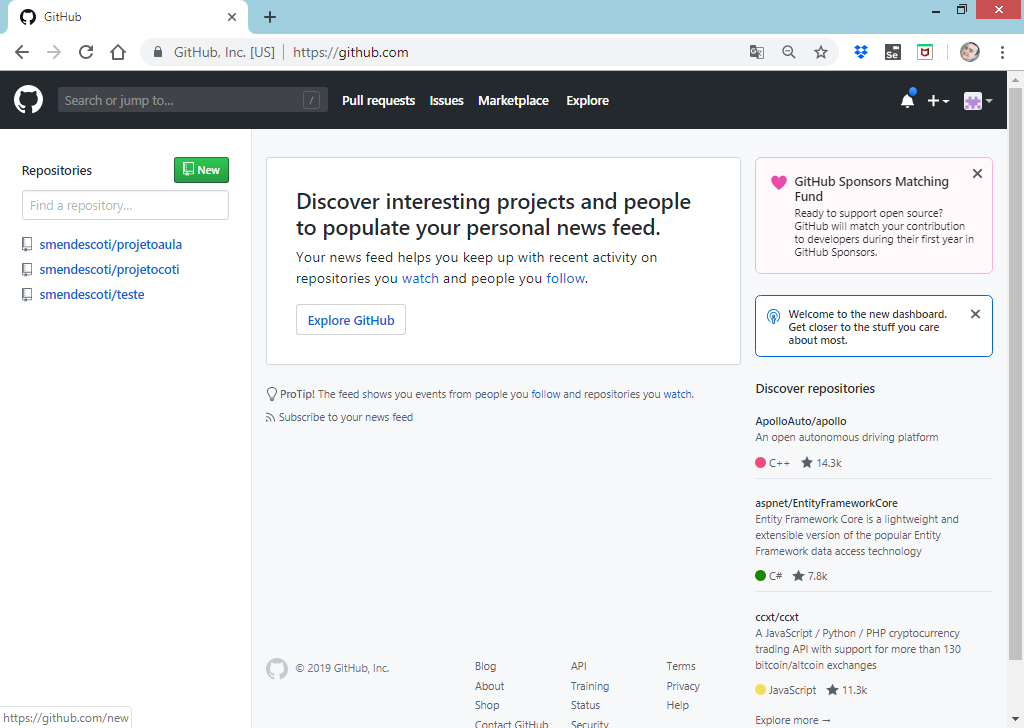
Criando uma nova conta:

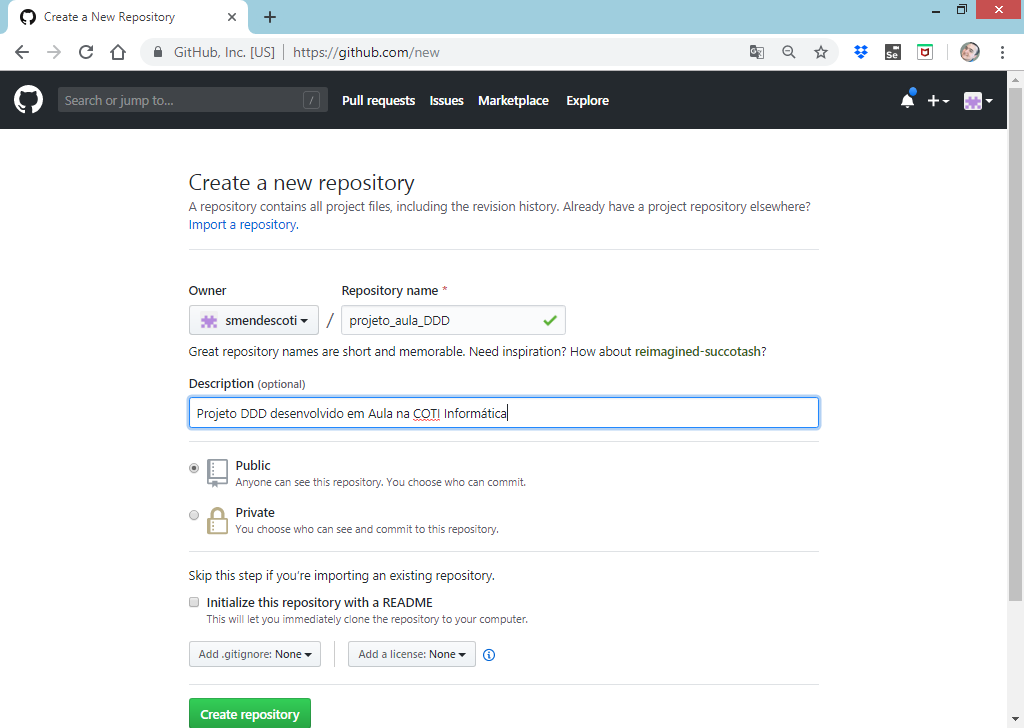


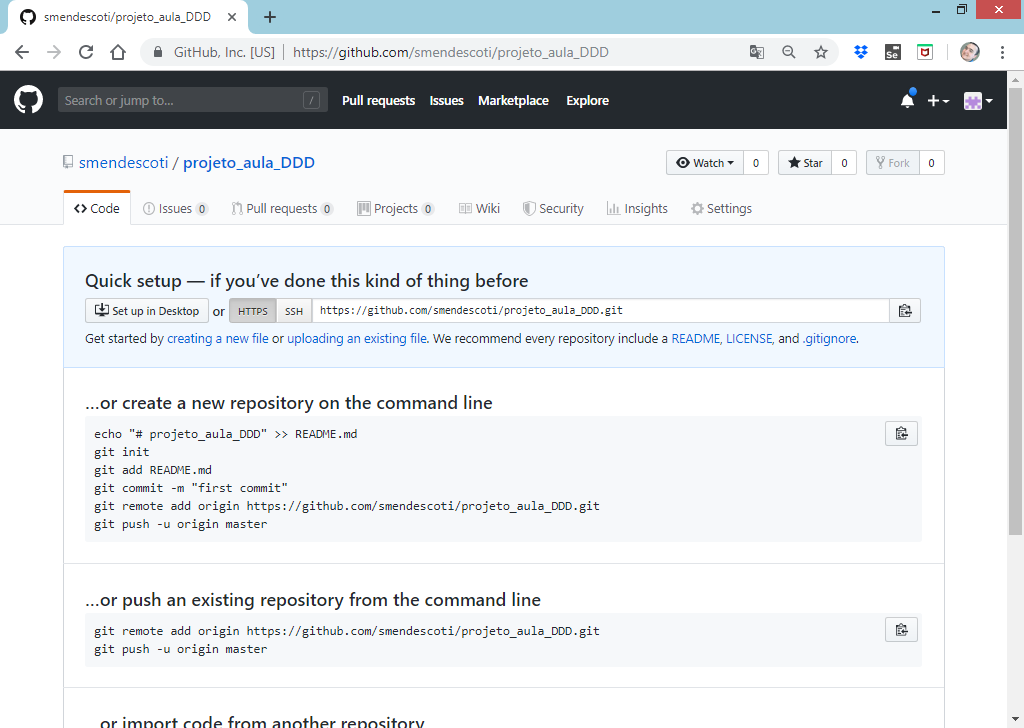
**Acessando a conta:**



**Criando um novo repositório de projeto:**





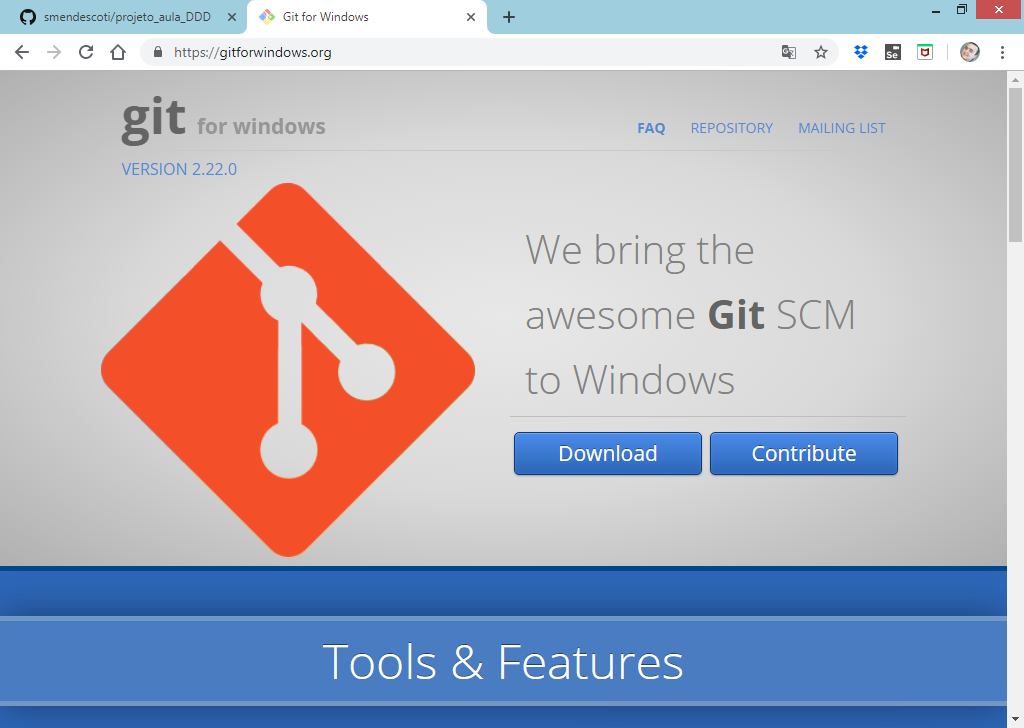


Endereço do repositório:

**https://github.com/smendescoti/projeto\_aula\_DDD.git**

Para que possamos fazer o upload do projeto para o GitHub iremos utilizar uma interface de comandos denominada **GitBash**

<https://gitforwindows.org/>



Registrando o usuário do GitHub

Prompt de comandos do GitBash

**COTI@COTI308MAQ16 MINGW64 ~**

**$ cd c:**

**COTI@COTI308MAQ16 MINGW64 /c**

**$ cd Aula26**

**COTI@COTI308MAQ16 MINGW64 /c/Aula26**

**$ git config --global user.name 'smendescoti'**

**COTI@COTI308MAQ16 MINGW64 /c/Aula26**

**$ git config --global user.email 'sergio.coti@gmail.com'**

Configurando o Repositório e subindo os arquivos

Prompt de comandos do GitBash

**COTI@COTI308MAQ16 MINGW64 /c/Aula26**

**$ git init**

**Initialized empty Git repository in C:/Aula26/.git/**

**COTI@COTI308MAQ16 MINGW64 /c/Aula26 (master)**

**$ git add Aula26.sln**

**COTI@COTI308MAQ16 MINGW64 /c/Aula26 (master)**

**$ git add Projeto.Presentation**

**COTI@COTI308MAQ16 MINGW64 /c/Aula26 (master)**

**$ git add Projeto.Application**

**COTI@COTI308MAQ16 MINGW64 /c/Aula26 (master)**

**$ git add Projeto.Domain**

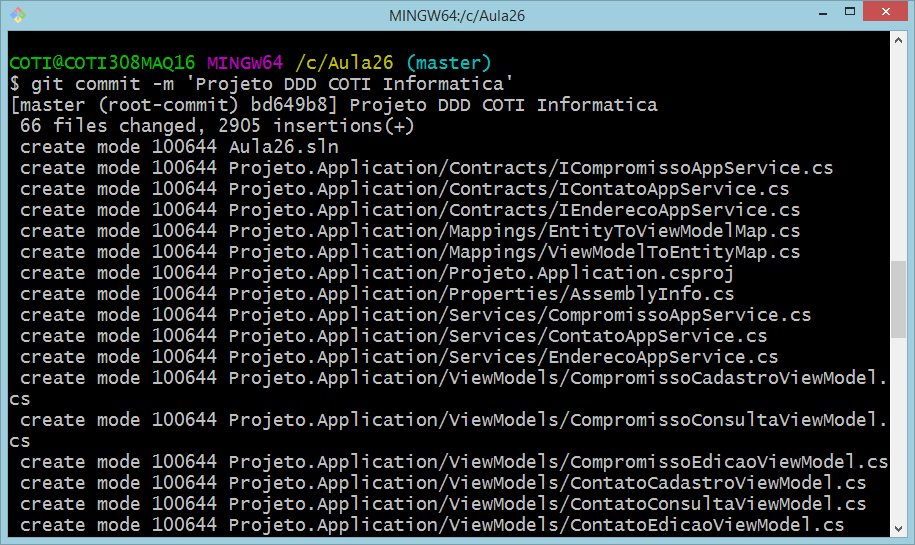
**COTI@COTI308MAQ16 MINGW64 /c/Aula26 (master)**

**$ git add Projeto.Infra.Data**

------------------------------------------------

**COTI@COTI308MAQ16 MINGW64 /c/Aula26 (master)**

**$ git commit -m 'Projeto DDD COTI Informatica'**

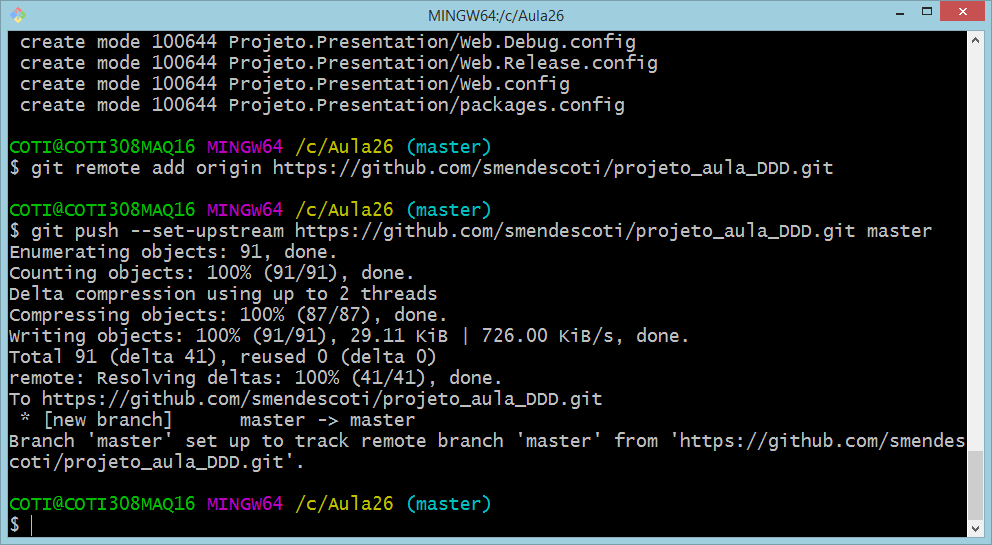


**COTI@COTI308MAQ16 MINGW64 /c/Aula26 (master)**

**$ git remote add origin https://github.com/smendescoti/projeto\_aula\_DDD.git**

**COTI@COTI308MAQ16 MINGW64 /c/Aula26 (master)**

**$ git push --set-upstream https://github.com/smendescoti/projeto\_aula\_DDD.git master**



**Código armazenado no GitHub:**

