


02 - Instalação VSCode + extensão TOTVS Developer Studio - LGX

 Tempo aproximado para leitura: superior a 15 minutos

INTRODUÇÃO

O editor de código-fonte **Visual Studio Code**, ou simplesmente **VSCode**, é ferramenta multiplataforma e amplamente utilizada para desenvolvimento com várias linguagens de programação através da instalação de extensões (plugins).

A **TOTVS** desenvolveu a extensão **TOTVS Developer Studio Code** para desenvolvimento nas linguagens **ADVPL**, **4GL** e **TLPP**, sendo que a linguagem **4GL**, utilizada para desenvolvimento do produto **LOGIX**, está disponível para uso desde o [início do 2o semestre de 2020](#).

A seguir serão apresentados tópicos resumidos envolvendo algumas dicas e procedimentos de instalação e configuração do **VSCode** e extensão **TDS for VSCode**, sendo que em alguns pontos serão apenas apontados alguns links de páginas oficiais existentes no site **GitHub** onde encontram-se as documentações oficiais da ferramenta.



PASSO A PASSO

> Instalação VSCode

Se você ainda não tem o editor **VSCode** instalado, acesse <https://code.visualstudio.com/>, faça o download para a plataforma desejada e instale conforme passo-a-passo do site oficial.

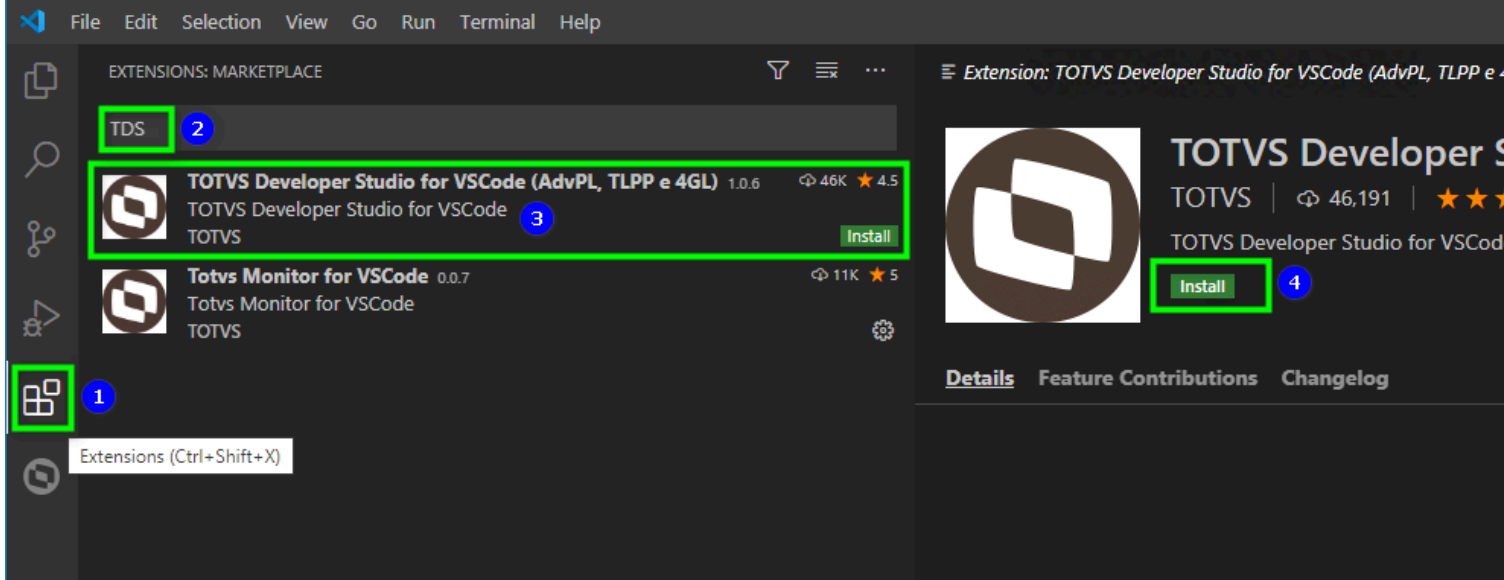


DICA

Na instalação Windows, o instalador sugere a pasta local do usuário por padrão e recomendamos manter esta sugestão, pois ao instalar na pasta de *Arquivos de Programas do Windows*, alguns controles de segurança do Windows podem impedir algumas atualizações futuras envolvendo **VSCode** ou até atualizações de extensões, dependendo do nível de segurança ativo no Windows.

> Instalação plugin TOTVS Developer Studio Code

A partir do editor **VSCode**, acompanhando a ilustração de itens na imagem abaixo, acesse na barra de ferramentas disponível na lateral esquerda na posição vertical, o atalho de extensões do VSCode, destacado no [item \(1\)](#) abaixo, informe "**TDS**" na caixa superior de filtro, [item \(2\)](#), selecione a extensão **TOTVS Developer Studio for VSCode (AdvPL, TLPP e 4GL)** conforme mostra o [item\(3\)](#) e clique em **INSTALL** [item \(4\)](#).



Para documentação completa sobre a extensão **TDS for VSCode** acesse <https://github.com/totvs/tds-vscode>, que contém o passo-a-passo de instalação e várias dicas de configuração.

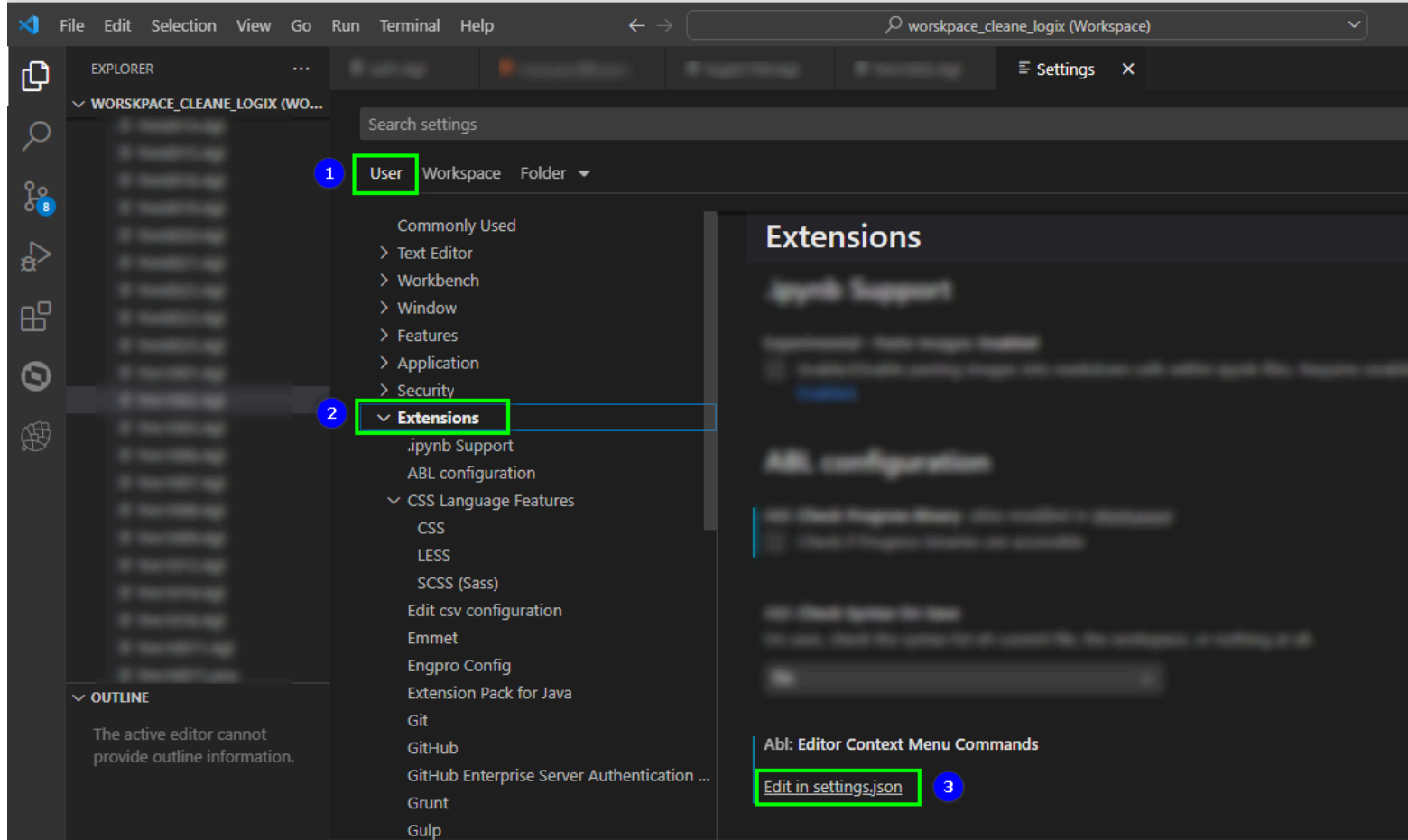
> Configurando o ENCODE (formato de edição) de arquivos no VSCode

IMPORTANTE

No Logix o formato de codificação padrão dos códigos fonte é **ISO8859-1** ou **CP1252** e para que isso seja reconhecido automaticamente no **VSCode** é preciso configurar a lista de extensões dos arquivos fontes previstos para cada linguagem, sendo que no Logix fazemos uso atualmente de 3 linguagens de desenvolvimento, sendo: **4GL**, **ADVPL** e **TLPP**, sendo esta última prevista a partir do build **Harpia**.

Siga o passo a passo abaixo e veja como é simples configurar as extensões de arquivos por linguagem no **VSCode** e depois indicar o formato de codificação padrão de cada extensão.

No **VSCode** acesse no menu **File | Preferences | Settings** (*Arquivo >> Preferências >> Configurações*) e, conforme mostra a imagem abaixo, acesse a aba **User** item (1), opção **Extensions** item (2) e a direita acesse o link descrito como **Edit in settings.json** item (3) para abrir o arquivo de configurações do usuário do **VSCode**.



Com o arquivo **settings.json** aberto para edição inclua as configurações abaixo, verificando sempre se as respectivas chaves já estão informadas, para que não sejam registradas de forma duplicada.

DICAS!



A chave **files.encoding** é comum já existir e neste caso basta ajustar o seu conteúdo para **utf8**, conforme mostra o quadro abaixo.



Informe este bloco no final do arquivo e cuide com o delimitador das chaves no arquivo **JSON** que precisa sempre **usar vírgula (,)** entre uma chave e outra.

Chaves a serem definidas no arquivo settings.json do usuário no VSCode

```
"files.encoding": "utf8",
"files.associations": {
  "*.ini": "advpl",
  "*.pro": "4gl",
  "*.4gl": "4gl",
  "*.cnv": "4gl",
  "*.xml": "4gl",
  "*.per": "4gl",
  "*.msg": "4gl",
  "*.prw": "advpl",
  "*.apw": "advpl",
  "*.prg": "advpl",
  "*.prx": "advpl",
  "*.tlpp": "advpl",
  "*.ppp": "advpl",
  "*.ppx": "advpl",
  "*.apl": "advpl",
  "*.aph": "advpl",
  "*.ahu": "advpl",
  "*.tres": "advpl",
  "*.res": "advpl",
  "*.txt": "advpl"
```

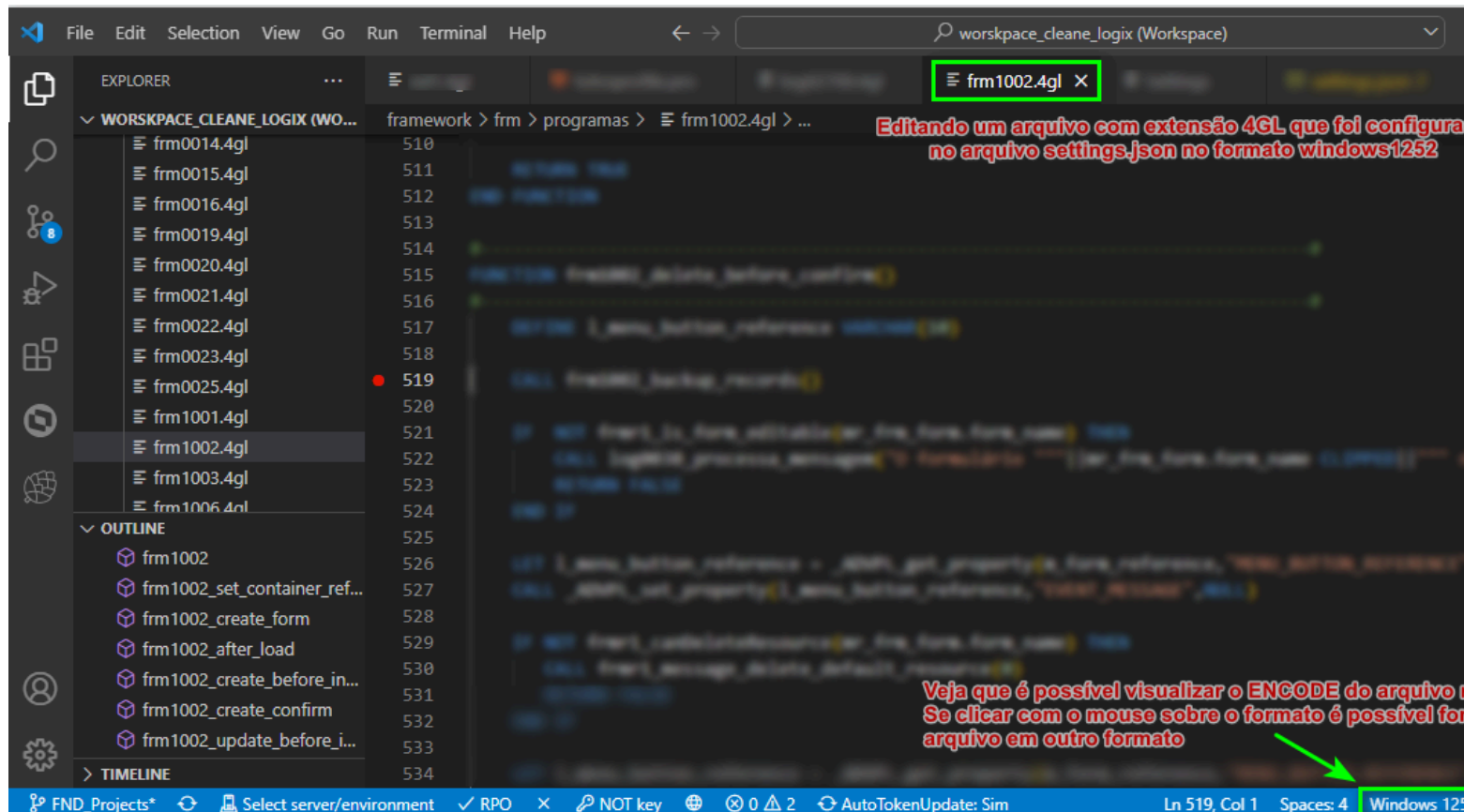
```

},
"[4gl]": {
  "files.encoding": "iso88591"
},
"[advpl]": {
  "files.encoding": "windows1252",
}

```

Salve o arquivo e a partir de agora seu editor **VSCode** já irá abrir cada fonte desenvolvido no **LOGIX** respeitando o formato de codificação (**ENCODE**) correto para não gerar problemas durante a execução do produto após compilação dos fontes, onde podem ser exibidas os textos em telas e mensagens contendo caracteres estranhos, por terem sido salvos com **ENCODE** incorreto.

Caso surja mais alguma extensão de arquivo que não esteja na lista acima, pode ser adicionada, desde que respeitando o formato e delimitadores das chaves e valores no arquivo **settings.json**.



> Criando nova workspace de desenvolvimento

Para criar um espaço de trabalho (**Workspace**) para desenvolvimento no **LOGIX**, a exemplo do **TDS (Totvs Developer Studio)** podíamos indicar uma pasta para armazenar todas as configurações desejadas, onde tínhamos um projeto com a lista de fontes em formato de árvore (**TreeView**) e também era possível configurar a lista de pastas de includes para o projeto (utilizados para compilação de fontes **ADVPL**).

Para o **VSCode** isso não muda, pois também é possível abrir tanto um único arquivo como uma pasta completa e no caso de abertura de uma pasta o **VSCode** entende que estamos definindo um novo espaço de trabalho (**Workspace**), onde então ele cria automaticamente uma subpasta chamada **.vscode** onde irá armazenar todos os arquivos das configurações utilizadas.

Toda vez que uma pasta nova é aberta a partir do **VSCode** e consequentemente uma nova **Workspace** é criada, por padrão será apresentada uma tela de **Boas Vindas** conforme imagem abaixo:



Welcome

Path to your SmartClient:

C:/totvs/bin/smartclient/smartclient.exe

Nenhum arquivo selecionado

Includes directory: ***Allow multiple directories**

Ex: C:/totvs/includes...

Nenhum arquivo selecionado //

* These settings can also be changed in
%HOME_USER%\.totvs\servers.json or
./vscode/launcher.json

Nesta tela você deverá:

Informar o atalho completo do Smartclient que será utilizado para execução.

Por padrão a extensão **TDS VSCODE** traz o atalho como "**c:/totvs/bin/smartclient/smartclient.exe**", mas basta acionar o botão

e selecionar o caminho correto do executável do smartclient que será utilizado.

É possível criar novos atalhos para Smartclients localizados em outras pastas quando for necessário, mas isso será citado em outro tópico adiante.

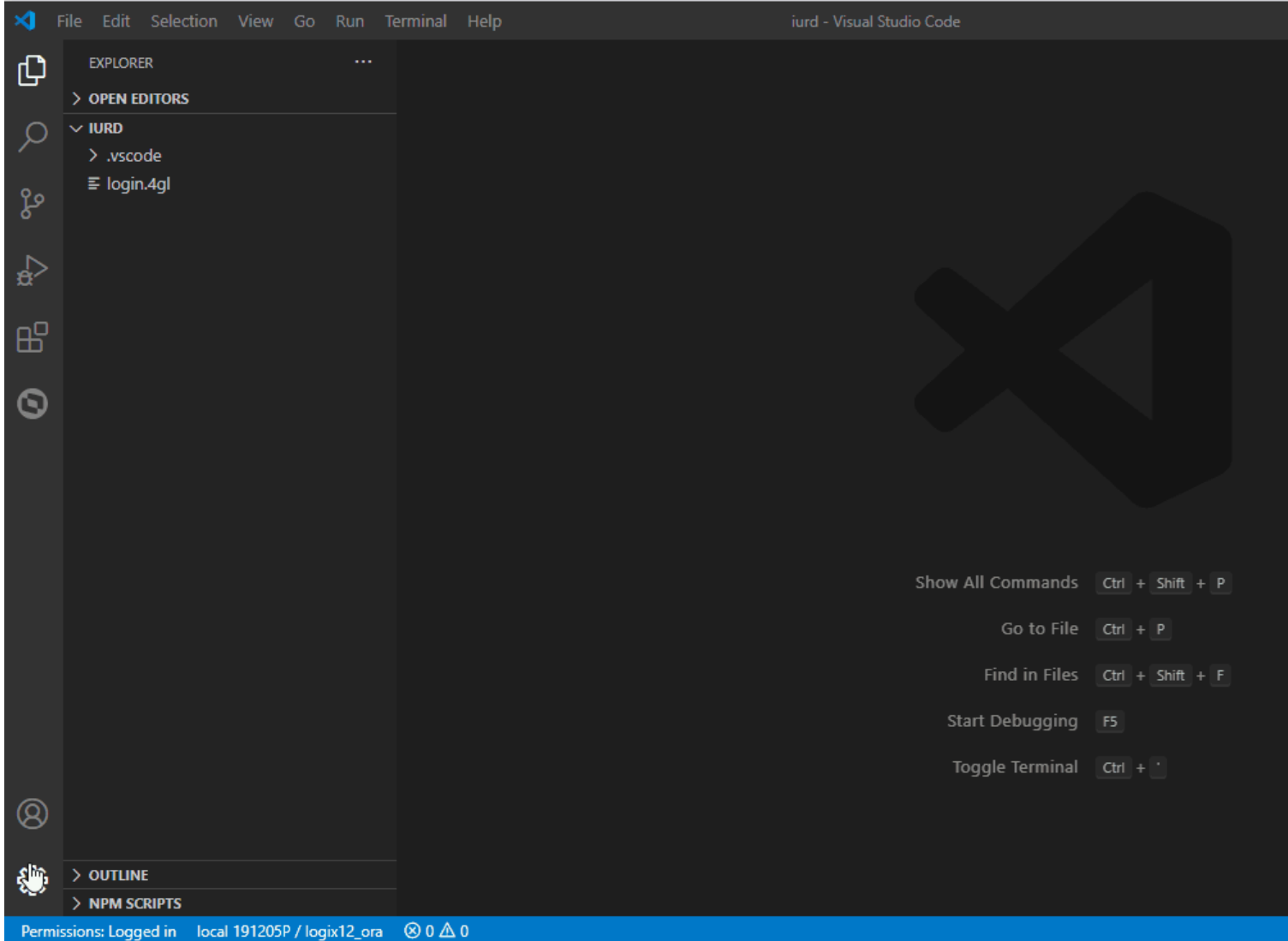
Informar as pastas de localização dos arquivos de INCLUDES.

Estas pastas são utilizadas para compilação de fontes desenvolvidos na linguagem **ADVPL**, sendo as pastas separadas por ponto e vírgula.

No caso de fontes 4GL não é necessário informar este campo.

A janela de Boas Vindas não aparece?

Para ativar ou desativar a exibição da janela de **Boas Vindas** é preciso mudar a configuração **Totvs Language Server: Welcome Page** (File | Preferences | Settings | Extensions | TOTVS | Welcome Page) conforme mostra o vídeo abaixo:

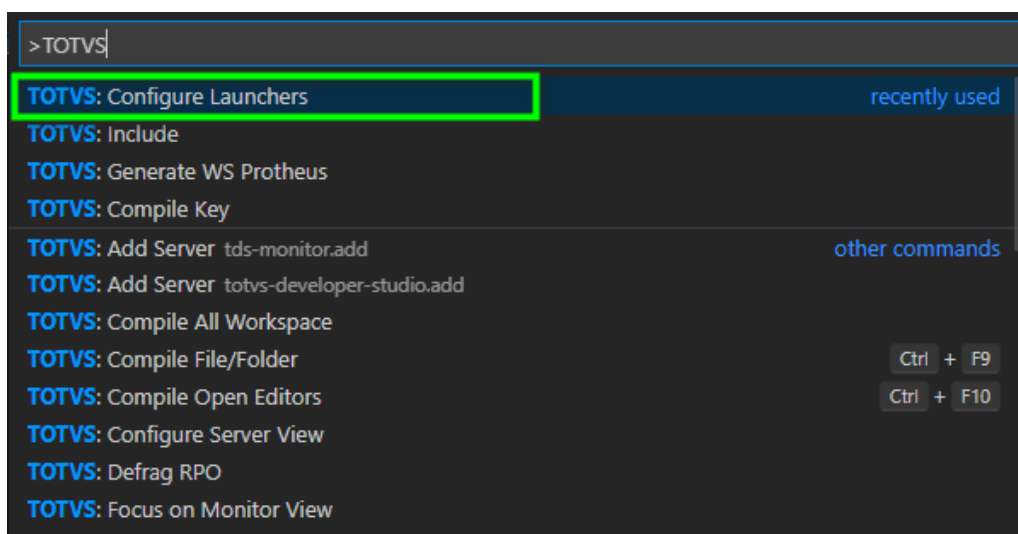


> Configurando atalhos do Smartclient (Launchers)

Para ajustar configurações de um atalho existente d do **Smartclient** é possível acionar a partir da lista de comandos no **VsCode** o comando **TOTVS:Configure Launchers**.

Pressione o atalho **CTRL+SHIFT+P** e informe o filtro do comando como "**TOTVS**" conforme imagem abaixo e selecione **TOTVS:Configure Launchers**

Selecione a opção **TOTVS:Configure Launchers**



Será apresentada esta janela para confi
(**Launcher**) para o Smartclient desejado

Launche

Choose launcher:

Program:

Arguments (-A):

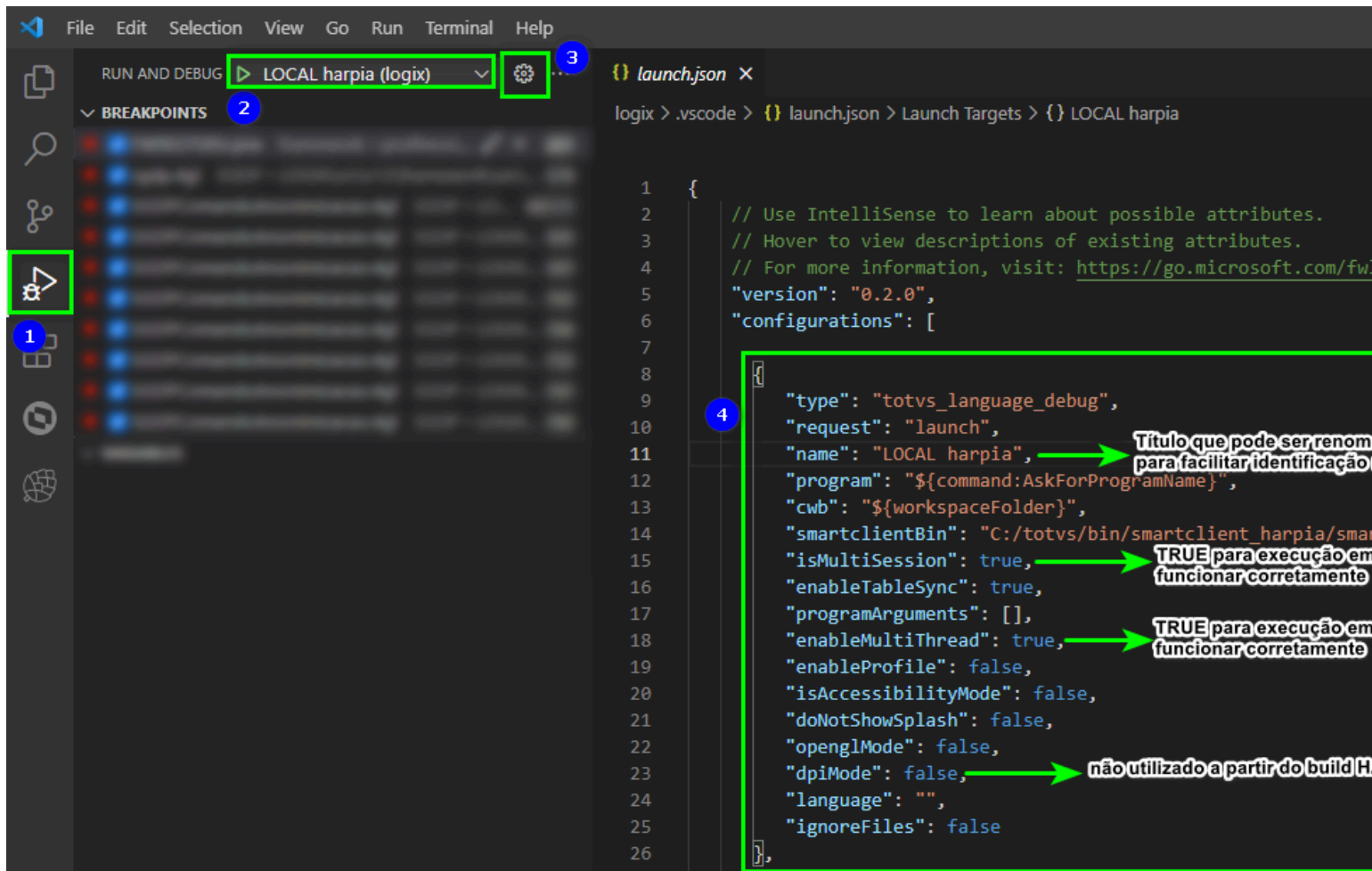
SmartClient:

Ex: C:/totvs/smartclient/smart

Nenhum a



É possível também registrar alterações dos atalhos do Smartclient diretamente no arquivo de configuração de forma manual no arquivo **/.vscode/launch.json** existente na pasta de workspace aberta no VsCode.

Para isso a partir do editor **VSCode**, acompanhando a ilustração de itens na imagem abaixo, acesse na barra de ferramentas disponível na lateral esquerda na posição vertical, o atalho de execução "**Run and Debug**", destacado no **item (1)** abaixo, em seguida selecione o atalho do smartclient desejado **item (2)** e em seguida clique sobre o botão de configurações ao lado **item (3)** para que seja possível abrir para edição o arquivo **launch.json** correto, já que para cada **Workspace** criada no **TDS-VSCode** existirá um arquivo **launch.json** diferente com a configuração de atalhos de **Smartclient**.



PRÉ-REQUISITOS ANTES DE CRIAR CONEXÃO COM APPSERVER NO VSCODE

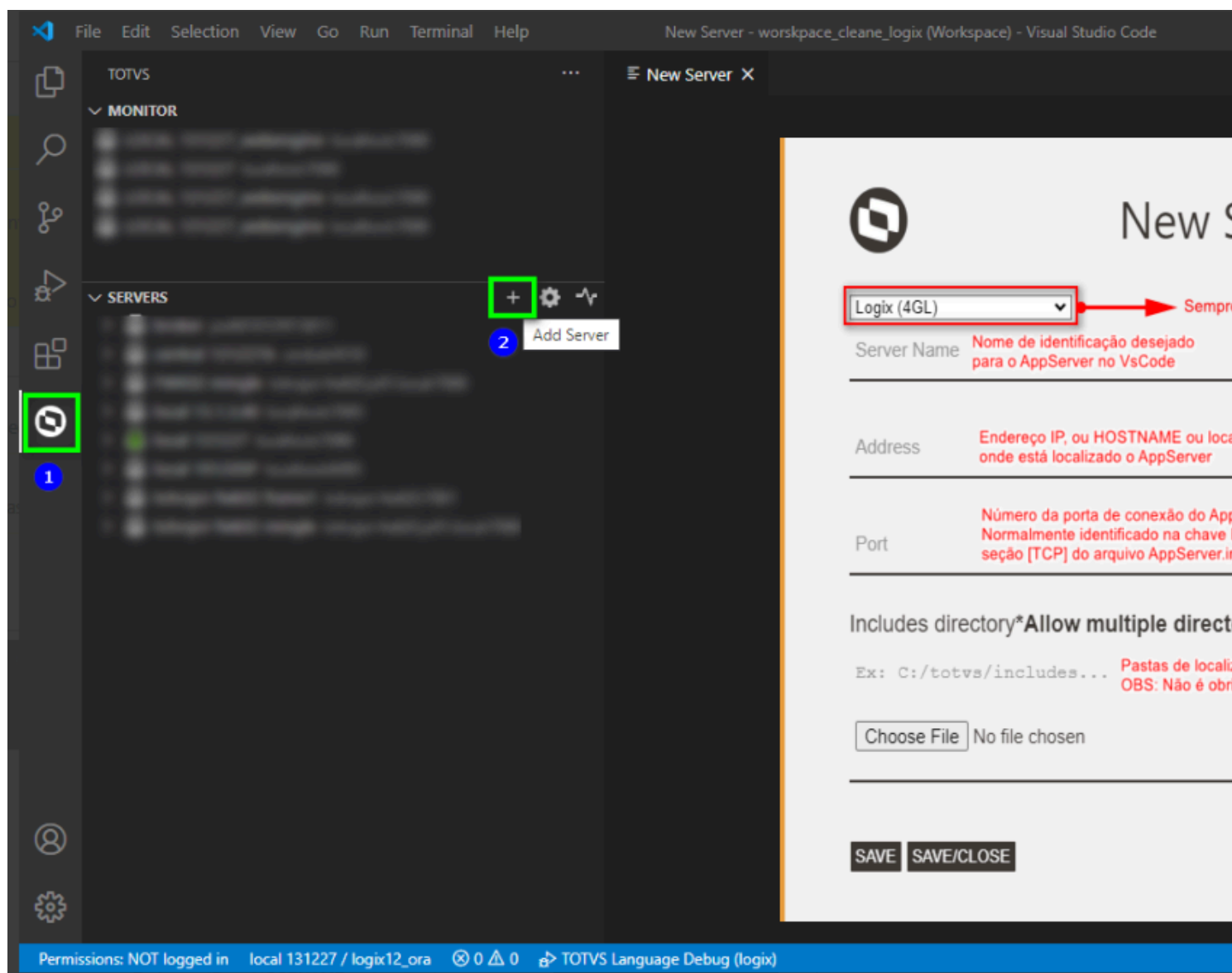
- **BUILD 7.00.131227A (32 bits)** → Utilizar revisão **13.1.3.53** ou superior
- **BUILD 7.00.210324P (64 bits - HARPIA)** → Utilizar revisão atualizada disponível para desenvolvimento **20.3.0.xx** ou superior
- **Extensão TDS-VSCode sempre atualizada** → Versão **1.3.12** ou superior
- Manter versão do **VSCode** atualizada
- Ter um **RPO Logix válido pacote 12.1.29 ou superior** já disponível na pasta configurada no ambiente Logix do arquivo **INI** do AppServer, pois será feita requisição de usuário e senha Logix para realizar autenticação e ter acesso ao **RPO**.

Para criar uma conexão com um novo **AppServer** para desenvolvimento no **LOGIX**, a partir do editor **VSCode**, acompanhando a ilustração de itens na imagem abaixo, acesse na barra de ferramentas disponível na lateral esquerda na posição vertical, o atalho **TOTVS AppServer** , destacado no **item (1)** abaixo, depois clique no na opção  na caixa superior de filtro, **item(2)**.



ATENÇÃO

NÃO DEIXE DE EXECUTAR TODAS INSTRUÇÕES ANTERIORES para que não tenha problemas na conexão com **AppServer**, ou seja, além de instalar o **VSCode** e a extensão **TDS for VSCode**, precisa já ter a **Workspace** definida para desenvolvimento.



> Conectando AppServer

Após conexão do AppServer já ter sido criada, deve-se registrar os ambientes de execução ou **Environment**, que já estão configurados no arquivo **INI** do **AppServer**.


Diferentemente de como os ambientes do **AppServer** eram acessados no **TDS**, onde ao conectar um **AppServer** já tínhamos disponíveis todos os ambientes para uso já registrados no **AppServer**, via **TDS for VSCode** deve-se registrar os ambientes um a um, baseados nos ambientes já registrados no arquivo **INI** do **AppServer**.

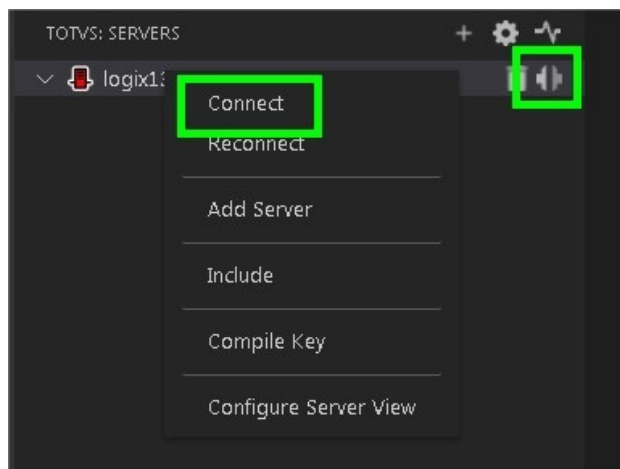
Não sabe o que é um Environment ou Ambiente no AppServer?

Um ambiente ou **Environment** é uma seção do arquivo **INI** do **AppServer** onde configuramos o acesso para execução de programas, indicando caminho do repositório de objetos (**RPO**), banco de dados a ser conectado, pasta raiz do AppServer (**ROOTPATH**), pasta inicial de trabalhos (**STARTPATH**), arquivo de **SCHEMA** de banco de dados utilizado na compilação de fontes **4GL** (arquivos com extensão .sch). Veja um exemplo abaixo de um ambiente de conexão no **AppServer** utilizado para execução do produto Logix:

```
[Logix12_oracle]
OUTPUTDIR=c:\totvs\logix\lst\
SOURCEPATH=c:\totvs\logix\apo
ROOTPATH=c:\totvs\logix\
STARTPATH=\totvs_data\
RPODB=SQL
RPOLANGUAGE=Portuguese
RPOVERSION=102
RPOPREFIX=lgx
SCROLLCursorsize=100000000
DBALIAS=lgx12_oracle
DBDATABASE=ORACLE
FGLDBPATH=C:\TOTVS\schema\teste
```

01. Clique com o botão direito do mouse sobre o nome da conexão de **AppServer** já configurado anteriormente e selecione a opção **CONNECT**, conforme mostra imagem abaixo. Pode-se também clicar no ícone lateral indicado

como , que tem o mesmo objetivo.



✓ DICA IMPORTANTE

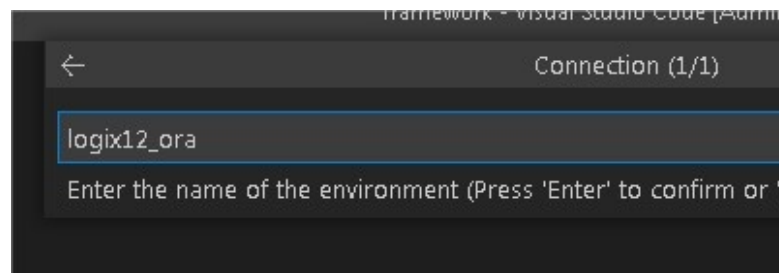
Para que a conexão e autenticação de usuário no ambiente do **AppServer** seja bem sucedido e seja possível realizar as compilações de fontes, extração de **PATCH** e aplicação de **PATCH**, já deve-se ter o **RPO Logix** disponível e válido devidamente configurado no AppServer.

Para não ter problemas de autenticação de usuário ao conectar um ambiente do **AppServer Logix**, utilize um **RPO Logix** oficial na versão 12.1.28 ou superior, pois existem pré-requisitos do **Framework** utilizados nesse processo de autenticação.

Para **RPOs** Logix anteriores a **12.1.28**, deve-se ao menos atualizar os pacotes **FIX Framework** disponíveis até a versão **12.1.27.FIX01**.

02. Durante o processo inicial de conexão, aguarde até que seja solicitado **AppServer** que deseja conectar.

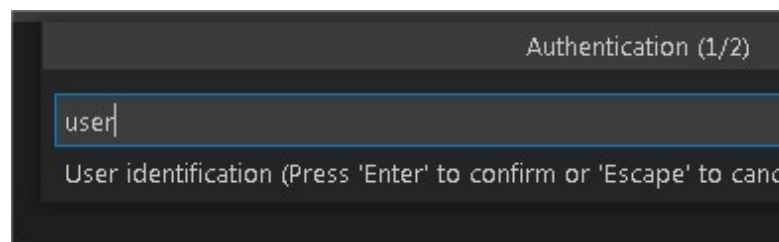
Na primeira conexão, como ainda não existe nenhum ambiente configurado, o sistema solicitará a criação de um novo ambiente. Se já houver ambientes já configurados no **INI do AppServer**, conforme imagem abaixo:



03. Informe um usuário Logix para a autenticação no ambiente.

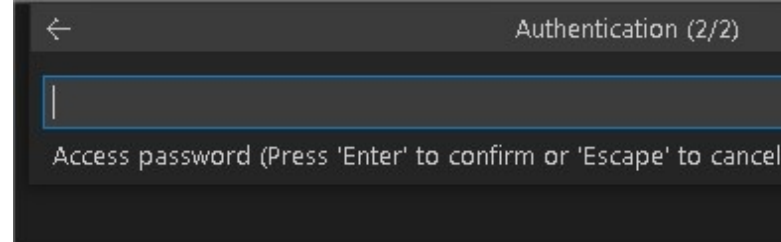
A partir do uso do **TDS for VSCode**, é exigida autenticação de usuário para obter a permissão para realizar as compilações de fontes, extração e aplicação de fontes.

No caso do Logix, ainda não existe um bloqueio no repositório de objetos, mas futuramente será implantado. Com isso pode-se informar um login de usuário como "user", mas deve ***** OBRIGATORIAMENTE ***** ser informado o login de usuário do banco Logix configurado para o ambiente que estiver sendo conectado.

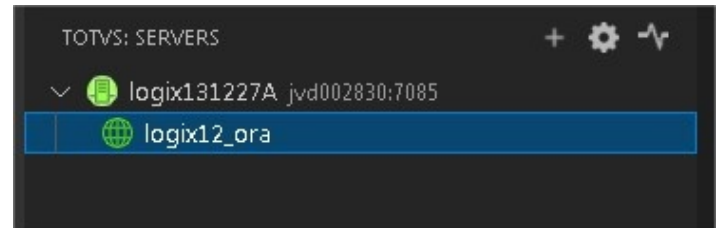


Esta obrigatoriedade de informar um login e senha de usuário Logix administrado pelo **12.1.2301**.

04. Informe a senha de usuário Logix (obrigatório)



05. Após a conclusão das etapas anteriores, o ambiente será conectado e apresentado na cor verde, como mostra imagem abaixo.



> Compilando programas

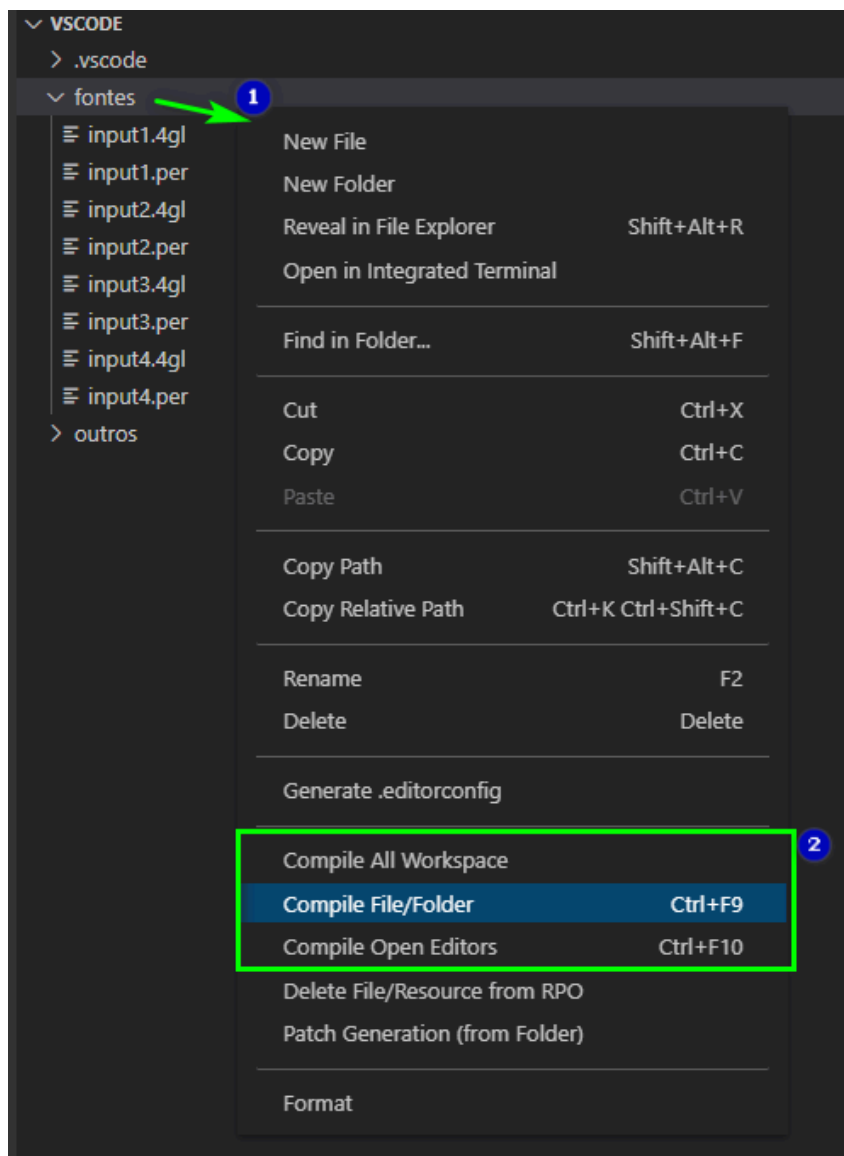
Para compilar fontes **ADVPL** e **4GL**, o processo é bem semelhante a antiga ferramenta de desenvolvimento **TDS**, no entanto no **VSCode** a compilação dos fontes **não irá mais atualizar o RPO padrão do produto, mas sim um RPO customizado**, que denominamos **RPO CUSTOM**. Para mais informações a respeito do conceito de **RPO CUSTOM** acesse [Instalação build HARPIA LOGIX](#).

Seguem algumas opções de compilação de fontes:

OPÇÃO 1

Compilação de fontes de uma pasta da Workspace em uso.

Clique com o botão direito do mouse sobre a pasta desejada **item (1)** e no menu popup apresentado selecione a opção **Compile File/Folder item (2)**.



OPÇÃO 2

Compilação avulsa de um fonte em edição.

Com o código fonte atualmente em edição e devidamente salvo, pode-se repetir o mesmo processo citado na **OPÇÃO 1**, no entanto o clique com o botão direito deverá ser realizado sobre o nome do fonte localizado na árvore de arquivos da **Workspace** e não sobre o nome de uma pasta.

Para um código em edição pode-se também pressionar a tecla de atalho **CTRL+F9**, como mostra a imagem da **OPÇÃO 1**,



Este processo de compilação de fontes de uma pasta (**OPÇÃO 1**) irá ignorar a compilação de fontes que forem detectados como já compilados no **RPO** e que não tenham sido detectadas alterações desde sua última compilação no **RPO**.

Para realizar a ação de **RECOMPILAÇÃO** (Recompile) de todos os fontes de uma pasta, **Workspace** ou dos fontes abertos para edição, conforme as opções do **item (2)** na imagem a esquerda, deve-se manter a tecla **SHIFT** pressionada e aí então clicar com o botão direito sobre a pasta desejada.


Veja que existem também as opções:

Compile All Workspace para compilar todos os fontes registrados em toda a **Workspace** em uso

Compile Open Editors para compilar todos os fontes atualmente abertos no editor.

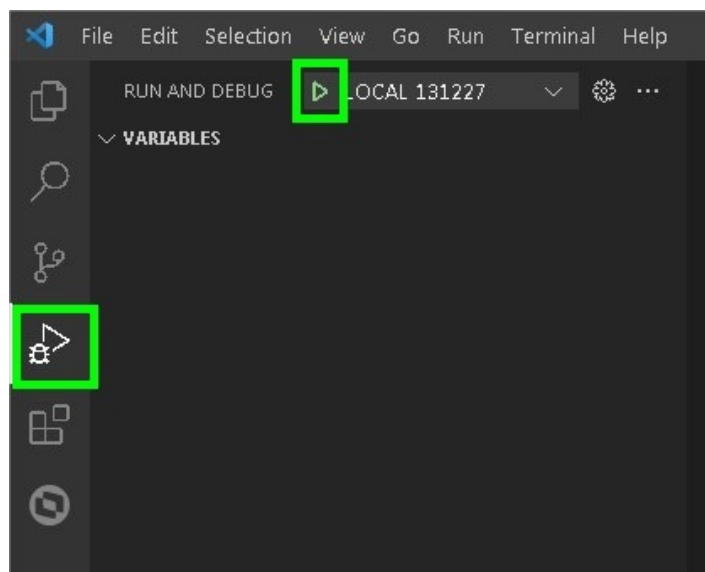
> Executando programas

Para executar um programa já compilado no **RPO** do ambiente/AppServer conectado é simples. Basta clicar na opção **Run and Debug (Ctrl +**

Shift + D), identificado pela imagem  na barra de ferramentas lateral, conforme mostra imagem do **PASSO 1**, selecionar o *launcher* desejado e clicar no botão **play**.

PASSO 1

Selecione o *launcher* desejado e clique no play



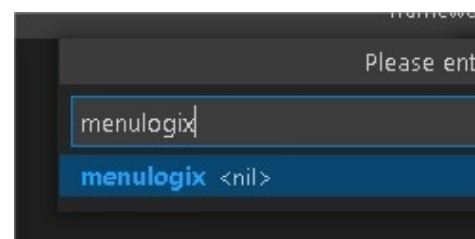
PASSO 2

O programa a ser executado dependerá das configurações do **launcher** (atalho para o **Smartclient**), onde é possível configurar o caminho do **TOTVS Smartclient** local que será utilizado para executar o programa no ambiente do **AppServer** conectado..

Para isto, acesse a seção anterior descrita como "Criando nova workspace de desenvolvimento" e veja como "registrar um novo atalho do SmartClient".

PASSO 3

Ao selecionar o atalho de **Smartclient** (Launcher) programa ou função já compilado no **RPO** que

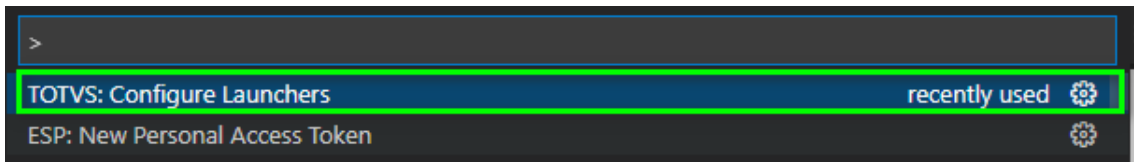


CONFIGURAÇÃO PARA PERMITIR DEBUG

Existem situações onde é necessário realizar o **DEBUG MultiThread** e isso é comum quando existem depuração de rotinas acionadas em paralelo, como por exemplo uma requisição **REST** a partir da ferramenta **POSTMAN**.

Neste caso é preciso que na configuração do atalho do **SmartClient (Launcher)** seja ativada a opção **Multiple Threads** e isso pode ser feito da seguinte forma:

- No **VSCode**, acione o atalho **CTRL+SHIFT+P -> Totvs: configure launchers**.



- Selecione o atalho do **Smartclient** já configurado (**launcher**) ou crie um novo conforme [item\(1\)](#) da imagem abaixo.
- Marque a opção **Enable Multi Threads** conforme [item\(2\)](#) da imagem abaixo.

Launcher Config

Launcher config name not defined.

Choose launcher:

1

Program:

Arguments (-A):

No arguments

Add Argument

Ask Arguments

SmartClient:

Ex: C:/totvs/smartclient/smartclient.exe...

Escolher arquivo

Nenhum arquivo selecionado

Opções:

2

☒ Enable multiple threads

☐ Enable Profile

Argumentos:

☒ (-M) Multiple sessions

☐ (-AC) Accessibility module

☐ (-Q) Don't display 'splash'

☐ (-OPENGL) Enable OpenGL mode

☐ (-DPI) Enable DPI mode

☐ Ignore files not found in WorkSpace (debugging)

Language (-L):

SAVE

3

SAVE/CLOSE



DICA

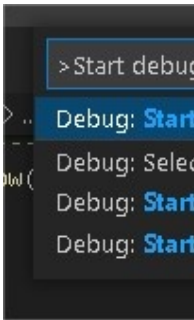
Para que possa informar o nome do programa ou function que deseja executar, sempre que for acionar este atalho do Smartclient, clique sobre o campo **Program** e escolha a opção `${command:AskForProgramName}`, pois ai toda vez que for debugar será perguntada qual rotina deseja executar, sempre sugerindo a última informada.

Para executar um programa em **modo DEBUG**, basta seguir os seguintes passos:

Localize a linha desejada nos fontes e clique somente uma vez ao lado da numeração de linhas do código fonte para inserir um ponto de **breakpoint**.



Execute a opção **S**



Ou apenas selecio

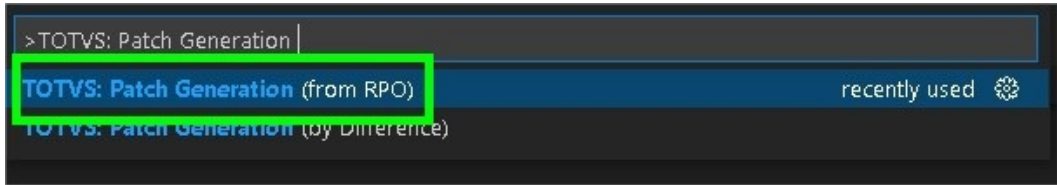


> Gerando ou aplicando PATCH de atualização

As etapas a seguir apresentam uma forma de geração e aplicação de **PATCH**.

Geração do PATCH

Execute a opção **TOTVS: Patch Generation (from RPO)**



Será apresenta
Utilize a opção
geração do PAT



☒ Ignore

LBROWS

LBROWS

LBROWS

Items: 2

Patch o

Ex: C:/to

Patch f

File nam

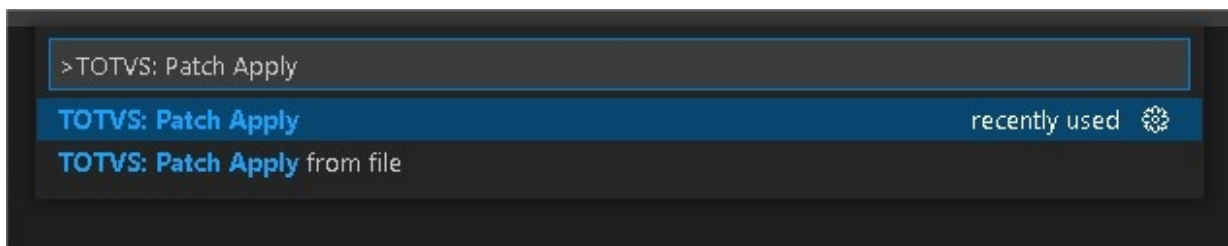
GENERA

* The g
fonts a

Configure os de
GENERATE/CL

Aplicando PATCH

Execute a opção **TOTVS: Patch Apply**



Será apresenta



Server

logix13

Address

jvd0028

Enviro

logix12

Patch

DELET

Patc

Showing

Escol

Apply

APPL

APPL

Uma janela de
Patch desejado

Após a seleção

> Instalando/Atualizando TDS for VSCode a partir de arquivo .VSIX no VSCode

As atualizações do **VSCode** podem ser baixadas manualmente do site **GITHUB** no link (<https://github.com/totvs/tds-vscode/releases/>),

preferencialmente procurando pela revisão descrita como

Latest

Na página de releases do **TDS for VSCode**, o arquivo com extensão **.vsix** é utilizado para ser instalação manual da extensão a partir do **VSCode** e é exibido na página de releases conforme exemplo na imagem abaixo:

▼ Assets 3



tds-vscode-1.3.5.vsix

126 MB




Source code (zip)

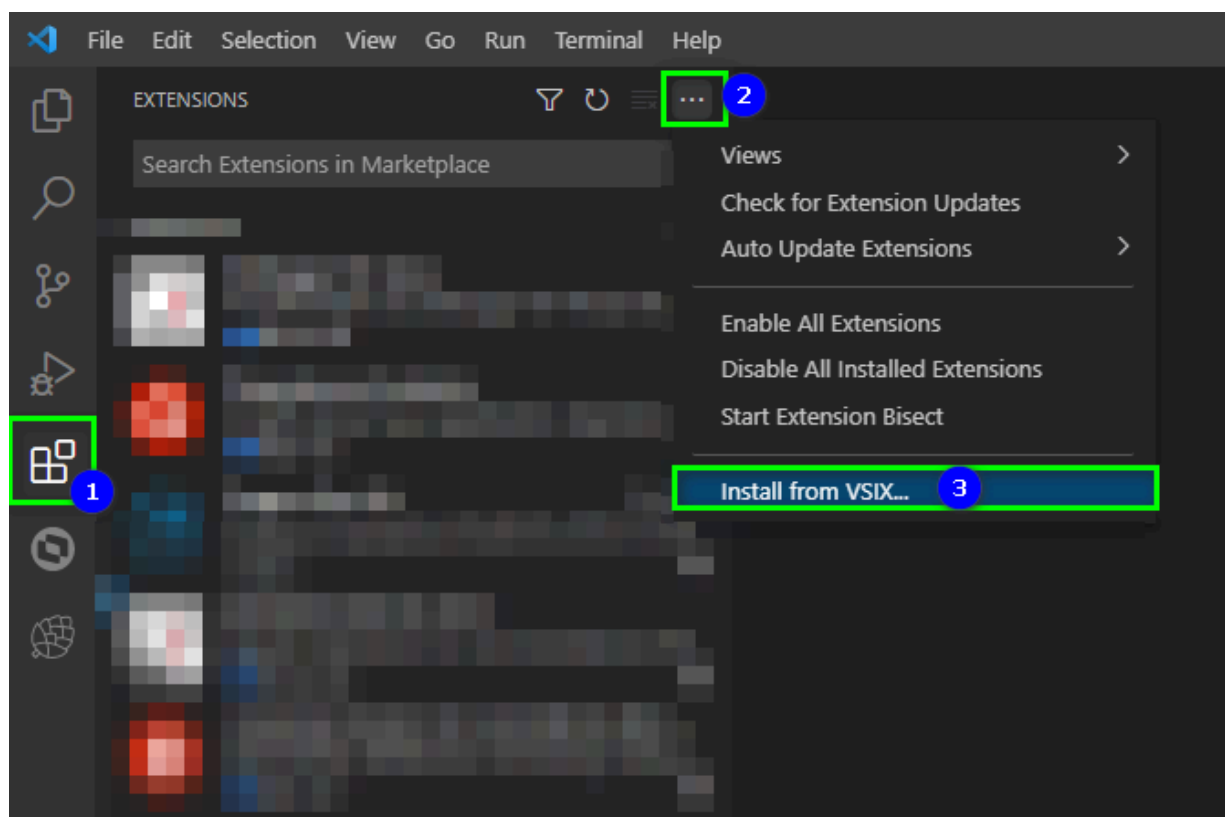


Source code (tar.gz)

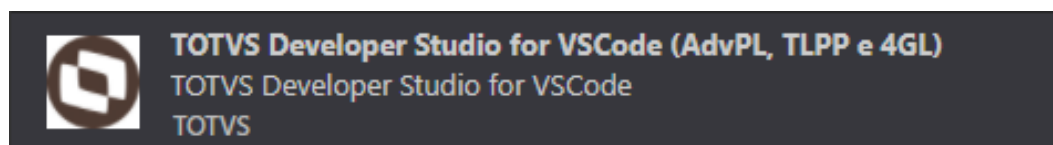


Baixe o arquivo **.VSIX** e instale-o a partir do **VSCode** utilizando o passo a passo a seguir:

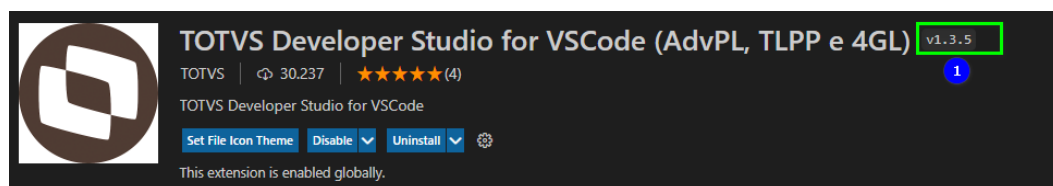
1. Acesse a lista de extensões (**CTRL+SHIFT+X**), ou clique na imagem identificada pelo [item\(1\)](#) na imagem abaixo.
2. A partir do menu das extensões identificado com  conforme indicado pelo [item\(2\)](#), escolha a opção [Install from VSIX...](#) [item\(3\)](#).
3. Escolha o arquivo de extensão **.VSIX** que foi baixado da página de releases do **TDS for VSCode** no site **GitHub** e confirme a instalação.



Para confirmar a versão da extensão instalada, basta clicar sobre a extensão **TDS for VSCode** que já estará sendo exibida na lista de extensões conforme imagem abaixo:



e na lateral direita conferir a versão da extensão conforme [item\(1\)](#) abaixo:



PRONTO! Sua extensão **TDS for VSCode** foi instalada/atualizada com sucesso. 😊

INFORMAÇÕES COMPLEMENTARES

Para saber mais detalhes de como instalar, configurar e obter dicas de uso do **TDS for VSCode**, acesse o link <https://marketplace.visualstudio.com/items?itemName=totvs.tds-vscode>

Este link além de apresentar uma idéia geral a respeito da extensão **TDS for VSCode**, demonstra o passo-a-passo para instalação e configuração de algumas funcionalidades e também dispõe de link de acesso para alguns vídeos que são recomendados para melhorar a familiarização com esta solução para desenvolvimento do produto **Logix**.

Para reportar problemas ou até sugerir melhorias relacionadas a extensão **TDS for VSCode**, acesse a aba **ISSUES** <https://github.com/totvs/tds-vscode/issues> e reporte a necessidade, seguindo as orientações descritas de como realizar abertura das novas **ISSUES**.



[Política de
privacidade](#)

[Termos
de uso](#)