C# Web developer test / exam

Abstract

In this job applicant exam, you will create a Postgres-database WEB application under MVC design pattern.In short, you will

1. Work with GitHub
2. Import the given Postgres export file (dump) into the locally installed Postgres DB.
3. Scaffold the database schema “ident” into the project Test01.Data
4. In the Test01.Web, you will create additional Views/Controllers/Models

Estimated time needed: 3-4 hours.

Exam takes two parts. First part should be completed at home (getting ready for the second part); second part is lead in the company premises.

Prerequisites

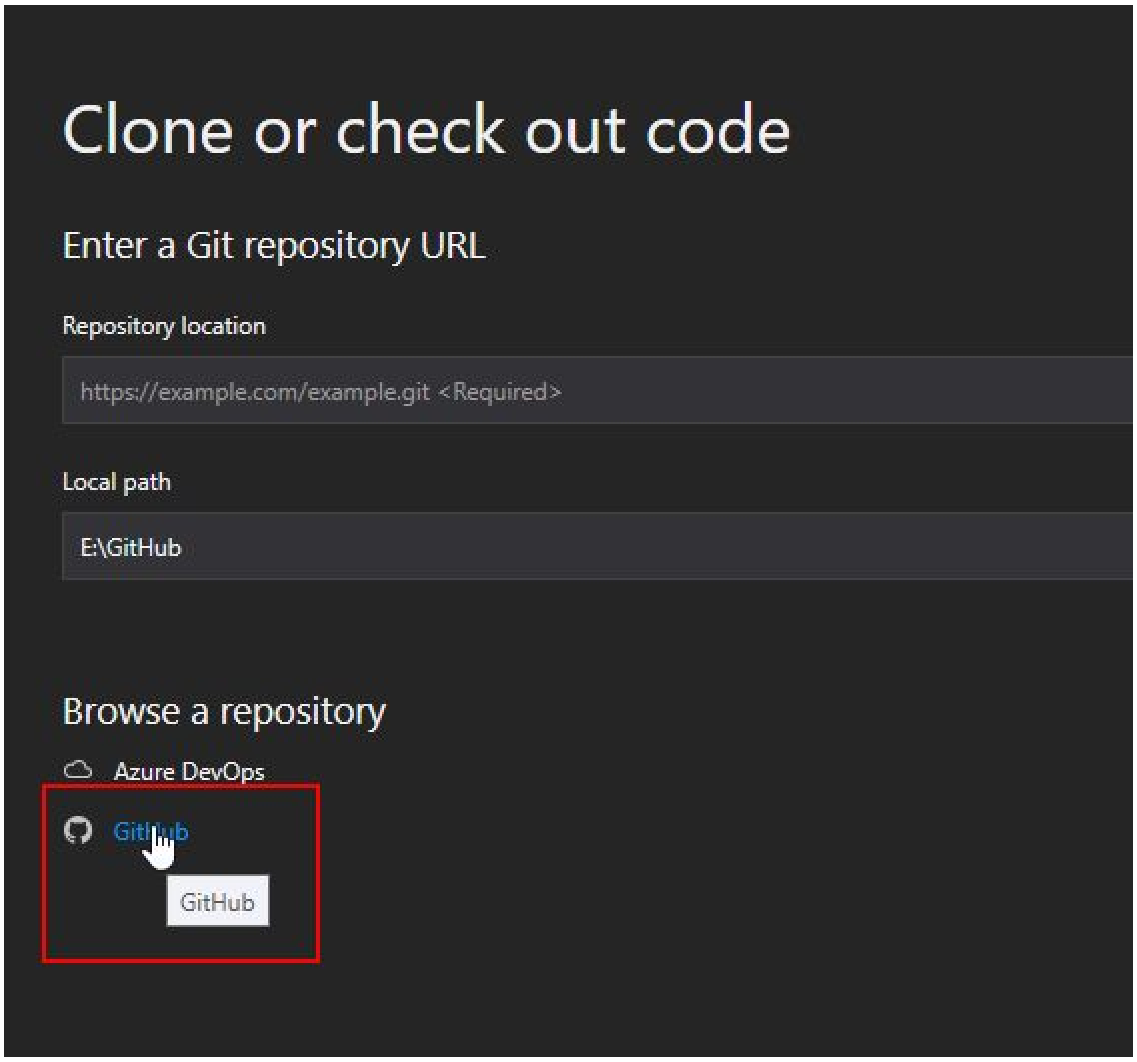
1. GitHub account
2. Visual Studio 2019 or later (recommended)
    a. Optionaly: Visual Studio Code
3. .NET Core 3.0
4. Local Postgres DB 10.4 with pgAdmin tool
    a. Installers

Good luck!

Exam steps - part one

Please measure time spent for completing this part. Write down time spent for each step.

1. **Checkout the github repository JobApplicants (You must be a registered user).**
   https://github.com/iSystemLabs01/JobApplicants
   You should Clone/Checkout the branch, named Candidate-xx (you will be given the xx number, ask your contact@iSystemLabs)



Clone or check out code

Enter a Git repository URL

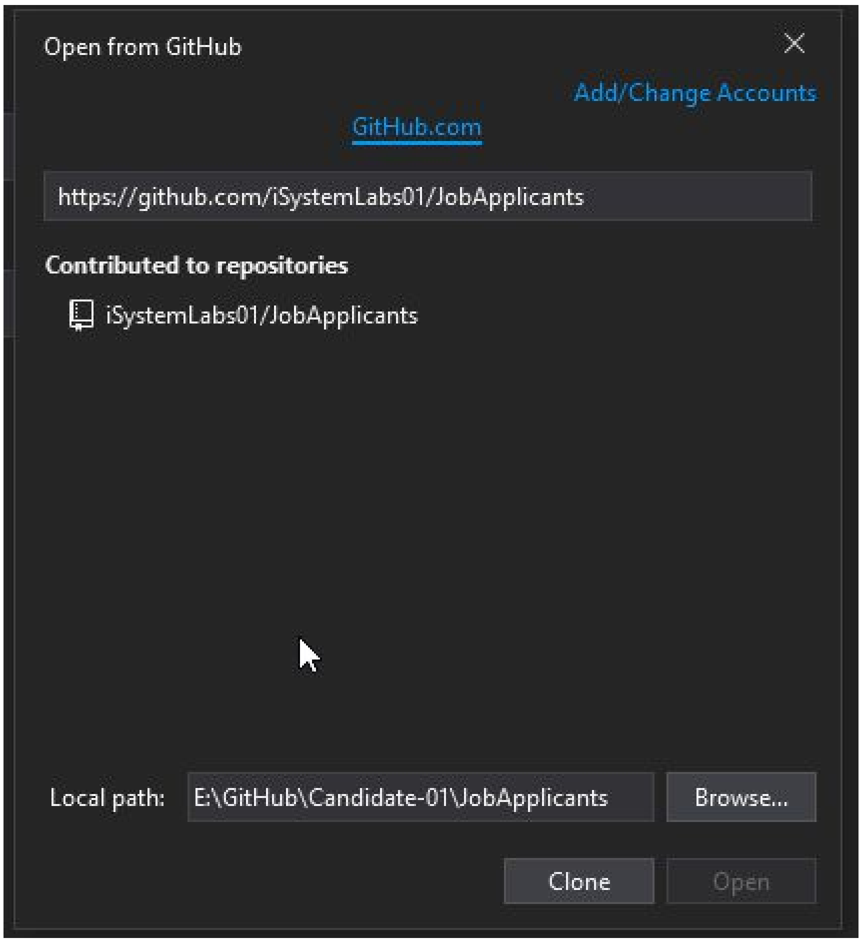
Repository location

https://example.com/example.git <Required>

Local path

E:\GitHub

Browse a repository


Open from GitHub

Add/Change Accounts

GitHub.com

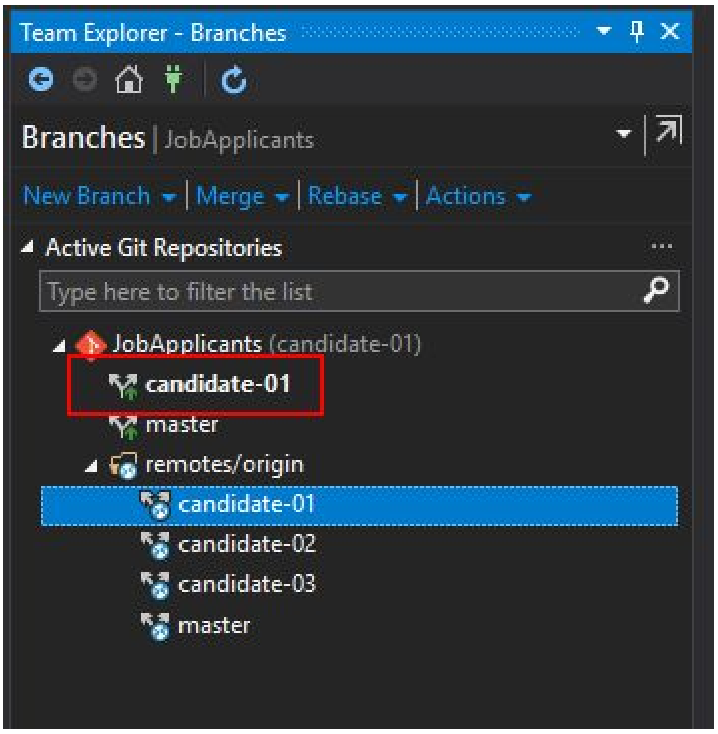
https://github.com/iSystemLabs01/JobApplicants

Contributed to repositories

iSystemLabs01/JobApplicants

Local path: E:\GitHub\Candidate-01\JobApplicants

Switch to your dedicated branch (candidate-xx)


Team Explorer - Branches

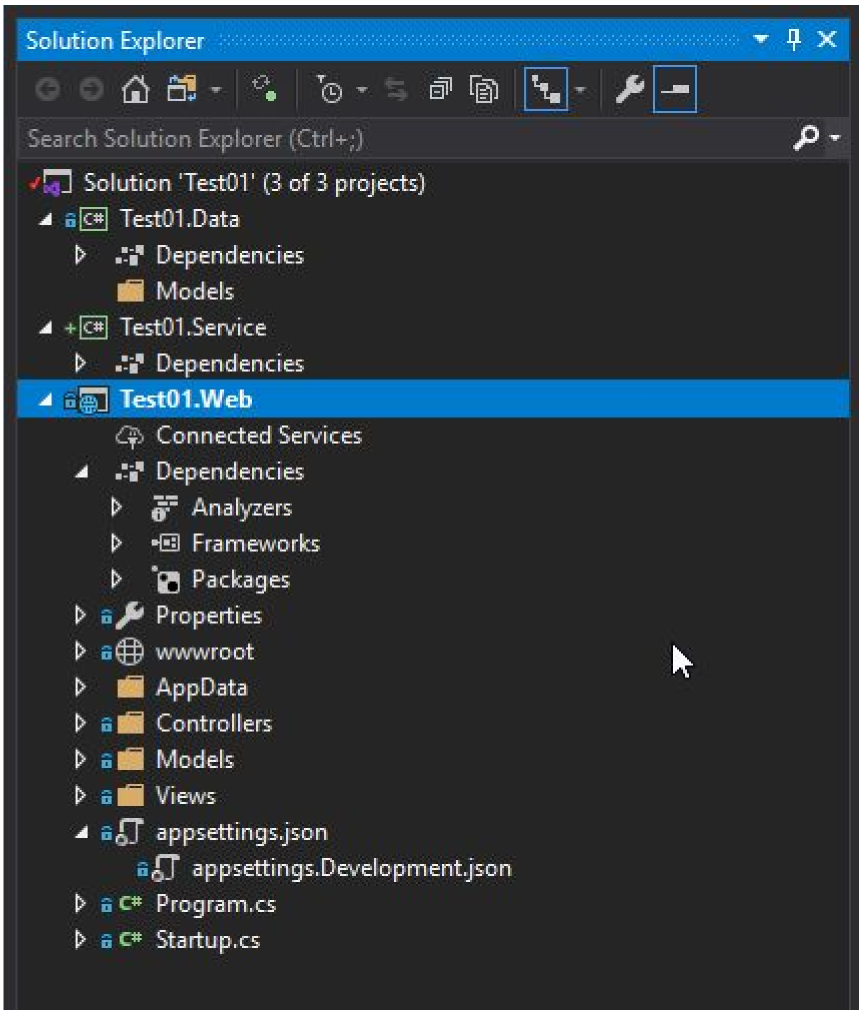
Branches | JobApplicants

Active Git Repositories

Type here to filter the list

```
JobApplicants (candidate-01)
    candidate-01
    master
  remotes/origin
      candidate-01
      candidate-02
      candidate-03
      master
```

Initial project structure looks like this:



2. **Setup the local Postgres DB instance (assuming you have installed it already)**
   a. Run pgAdmin

   b. Create User TrinityDBOwner (run this SQL script) You may change the password later.

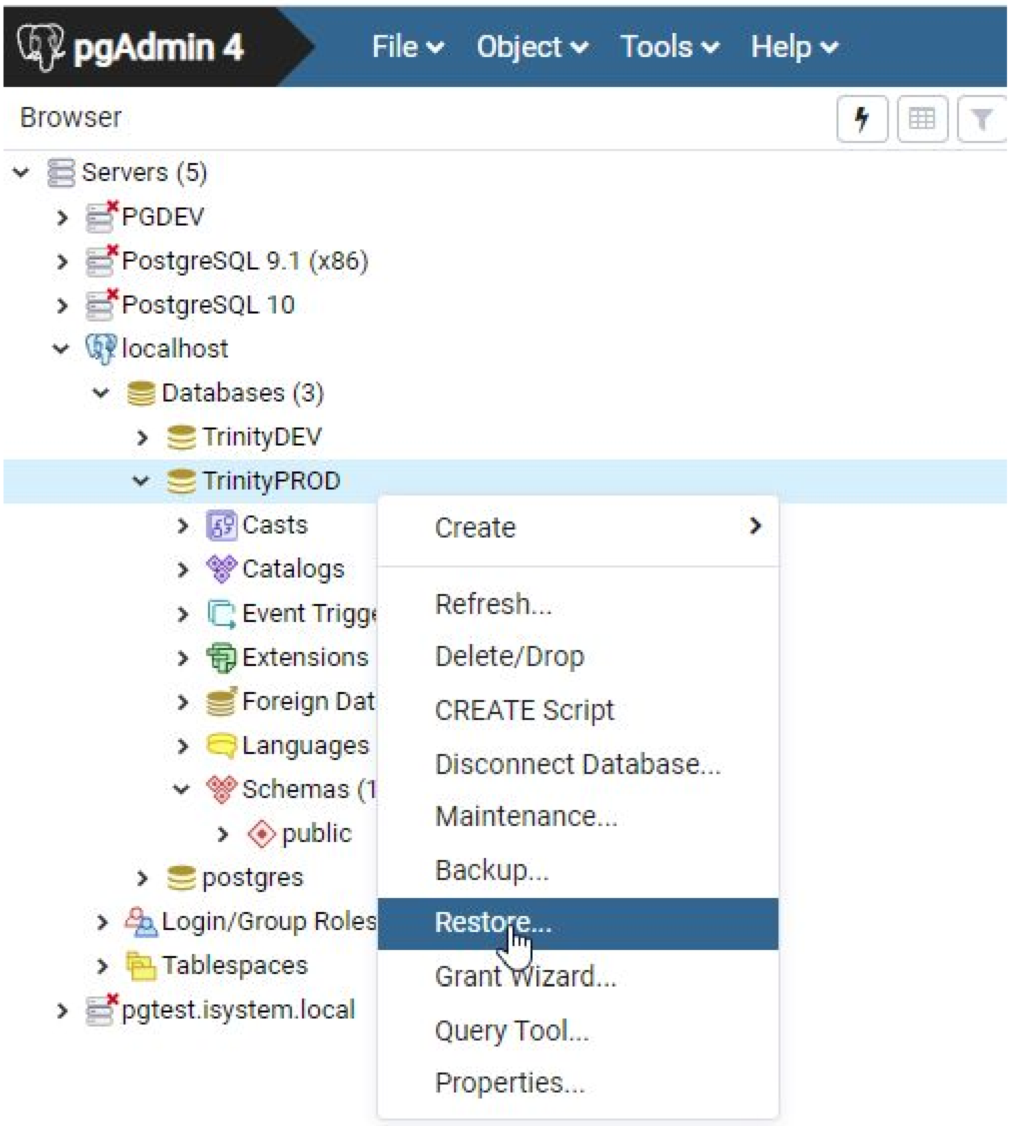
```
CREATE USER "TrinityDBOwner" WITH
      LOGIN ENCRYPTED PASSWORD 'md5f007eccc32160e67c81efc7d4b2984e2'
      SUPERUSER
      CREATEDB
      CREATEROLE
      INHERIT
      REPLICATION
      CONNECTION LIMIT -1;
```

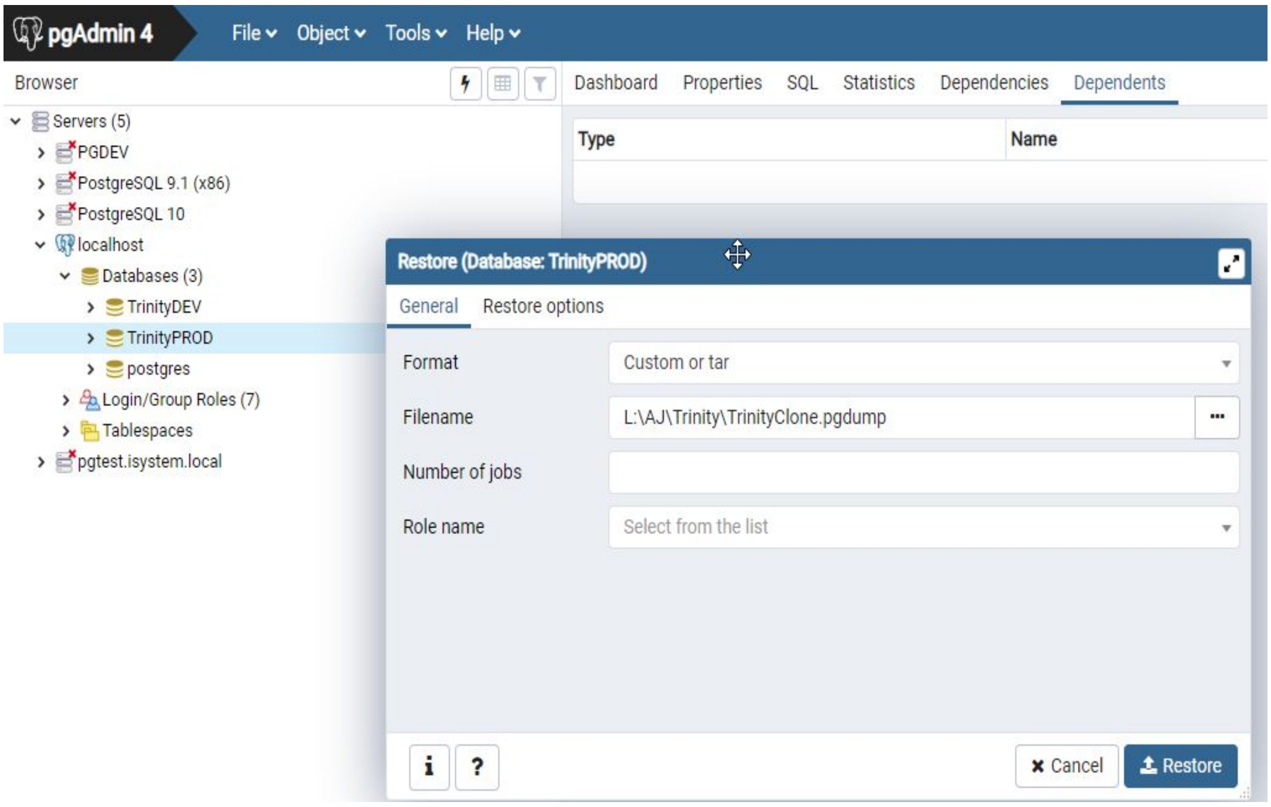
   c. Create Database TrinityPROD (run this script):

```
CREATE DATABASE "TrinityPROD"
    WITH
    OWNER = "TrinityDBOwner"
    ENCODING = 'UTF8'
    CONNECTION LIMIT = -1;

GRANT ALL ON DATABASE "TrinityPROD" TO PUBLIC;
```

d. Restore database TrinityPROD from TrinityClone.dump





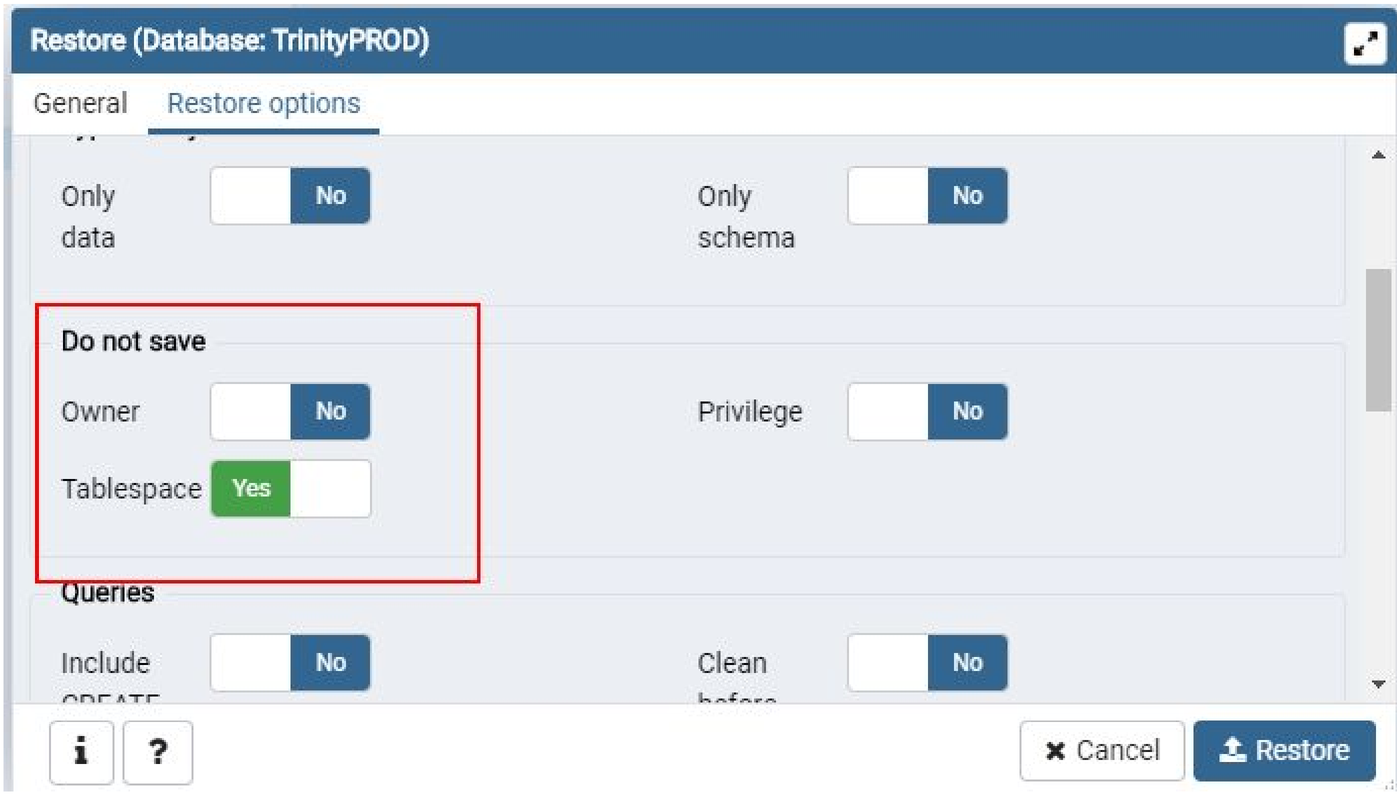
Format: Custom or tar

Filename: L:\AJ\Trinity\TrinityClone.pgdump

Number of jobs:

Role name: Select from the list

Exclude tablespace restore to avoid warnings / errors



Only data: No

Only schema: No

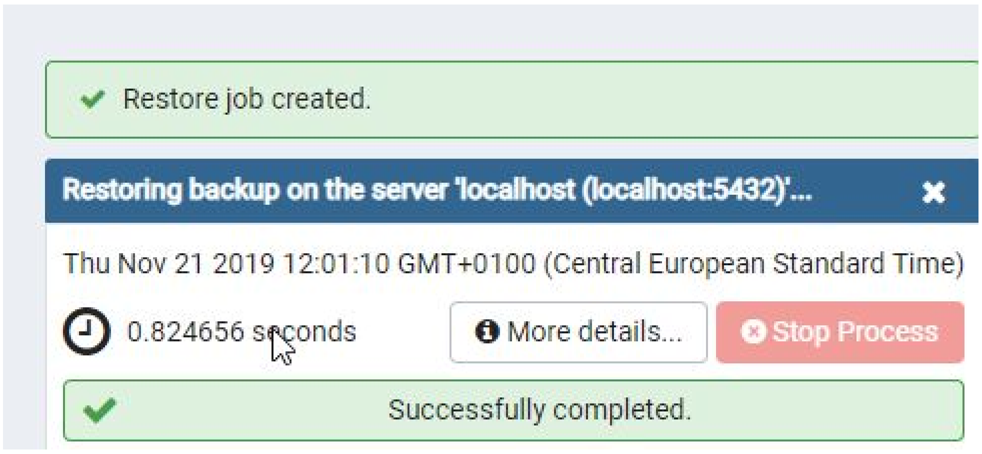
Do not save

Owner: No

Privilege: No

Tablespace: Yes

You should get:



Restore job created.

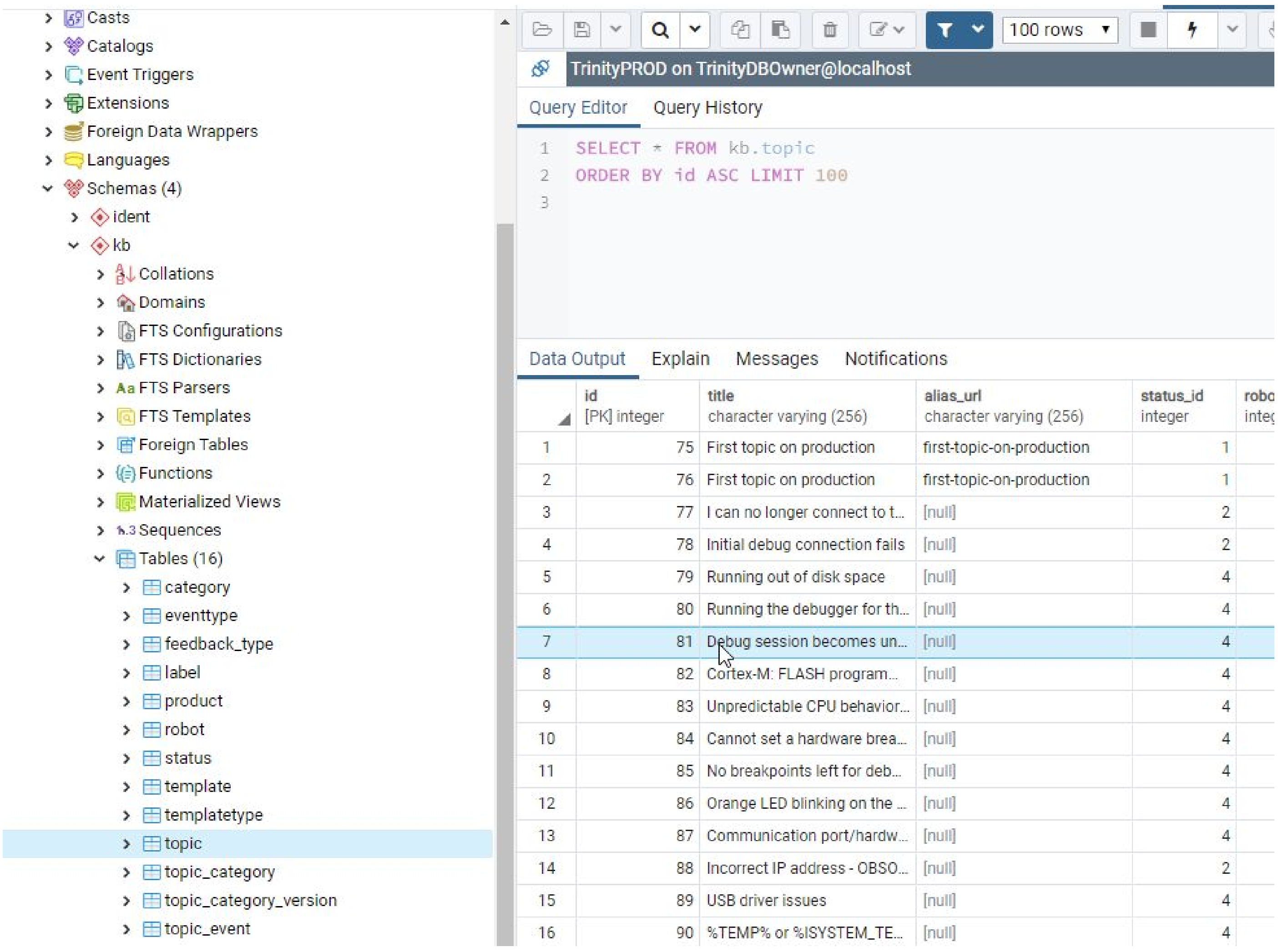
Restoring backup on the server 'localhost (localhost:5432)'...

Thu Nov 21 2019 12:01:10 GMT+0100 (Central European Standard Time)

0.824656 seconds

Successfully completed.

3. **Using pgAdmin, browse the restored database for data. Check table kb.topic**



TrinityPROD on TrinityDBOwner@localhost

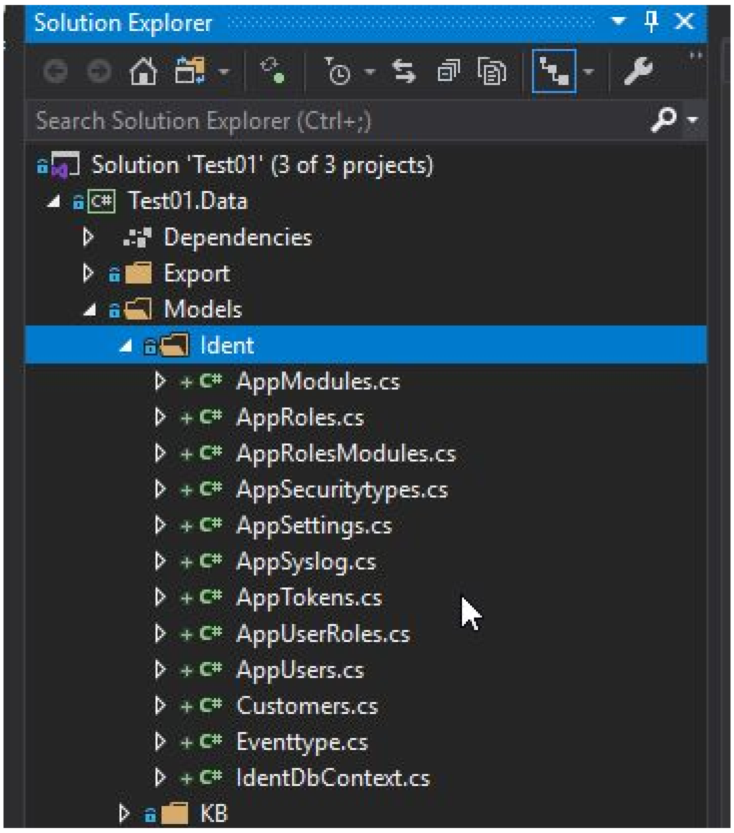
```
SELECT * FROM kb.topic
ORDER BY id ASC LIMIT 100
```

	id [PK] integer	title character varying (256)	alias_url character varying (256)	status_id integer	robo integ
1	75	First topic on production	first-topic-on-production	1	
2	76	First topic on production	first-topic-on-production	1	
3	77	I can no longer connect to t...	[null]	2	
4	78	Initial debug connection fails	[null]	2	
5	79	Running out of disk space	[null]	4	
6	80	Running the debugger for th...	[null]	4	
7	81	Debug session becomes un...	[null]	4	
8	82	Cortex-M: FLASH program...	[null]	4	
9	83	Unpredictable CPU behavior...	[null]	4	
10	84	Cannot set a hardware brea...	[null]	4	
11	85	No breakpoints left for deb...	[null]	4	
12	86	Orange LED blinking on the ...	[null]	4	
13	87	Communication port/hardw...	[null]	4	
14	88	Incorrect IP address - OBSO...	[null]	2	
15	89	USB driver issues	[null]	4	
16	90	%TEMP% or %ISYSTEM_TE...	[null]	4	

4. **Scaffold database schema “ident” into the project Test01.Data, folder Ident.**
   a. Use Entity Framework tool ef.
      Hint:

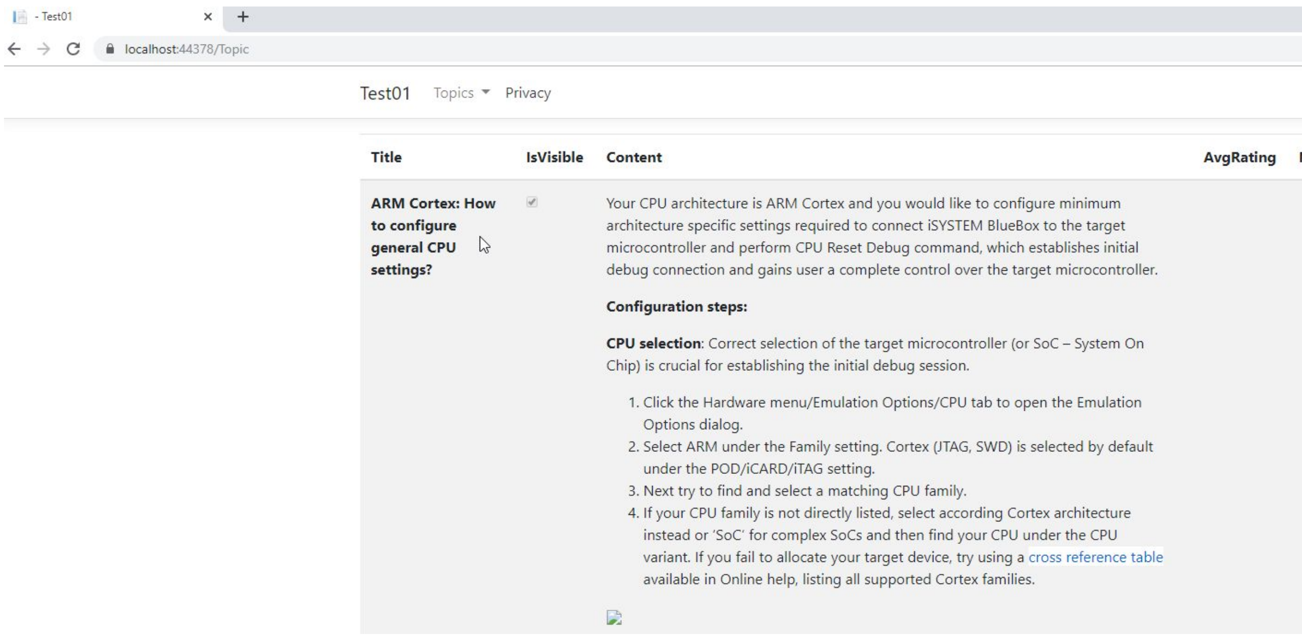
```
 dotnet ef dbcontext scaffold
"Host=localhost;Database=TrinityPROD;Username=TrinityDBOwner;Password=askmeonce"
Npgsql.EntityFrameworkCore.PostgreSQL -c IdentDbContext -o Models\Ident --schema ident
--force --verbose
```

Result:



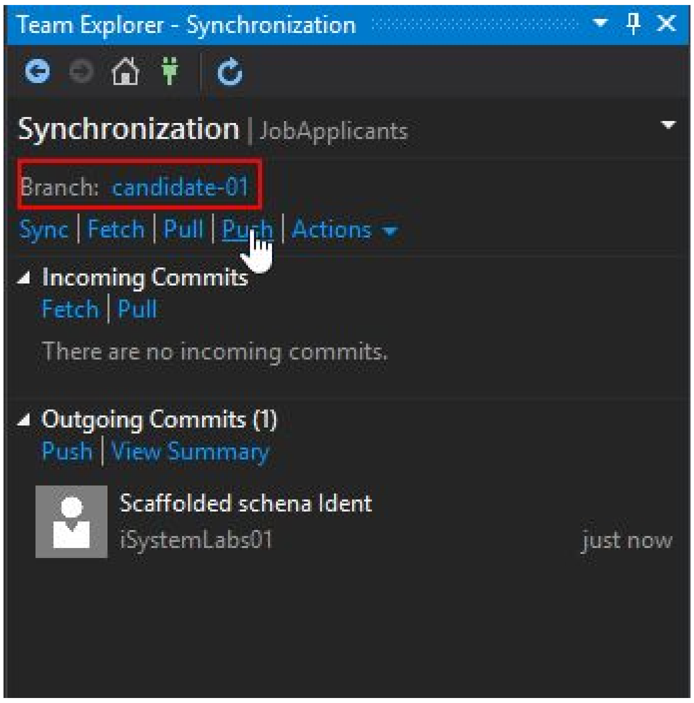
```
Solution 'Test01' (3 of 3 projects)
  Test01.Data
    Dependencies
    Export
    Models
      Ident
        AppModules.cs
        AppRoles.cs
        AppRolesModules.cs
        AppSecuritytypes.cs
        AppSettings.cs
        AppSyslog.cs
        AppTokens.cs
        AppUserRoles.cs
        AppUsers.cs
        Customers.cs
        Eventtype.cs
        IdentDbContext.cs
      KB
```

b. Build the project and check if it runs correctly. Check Topics index

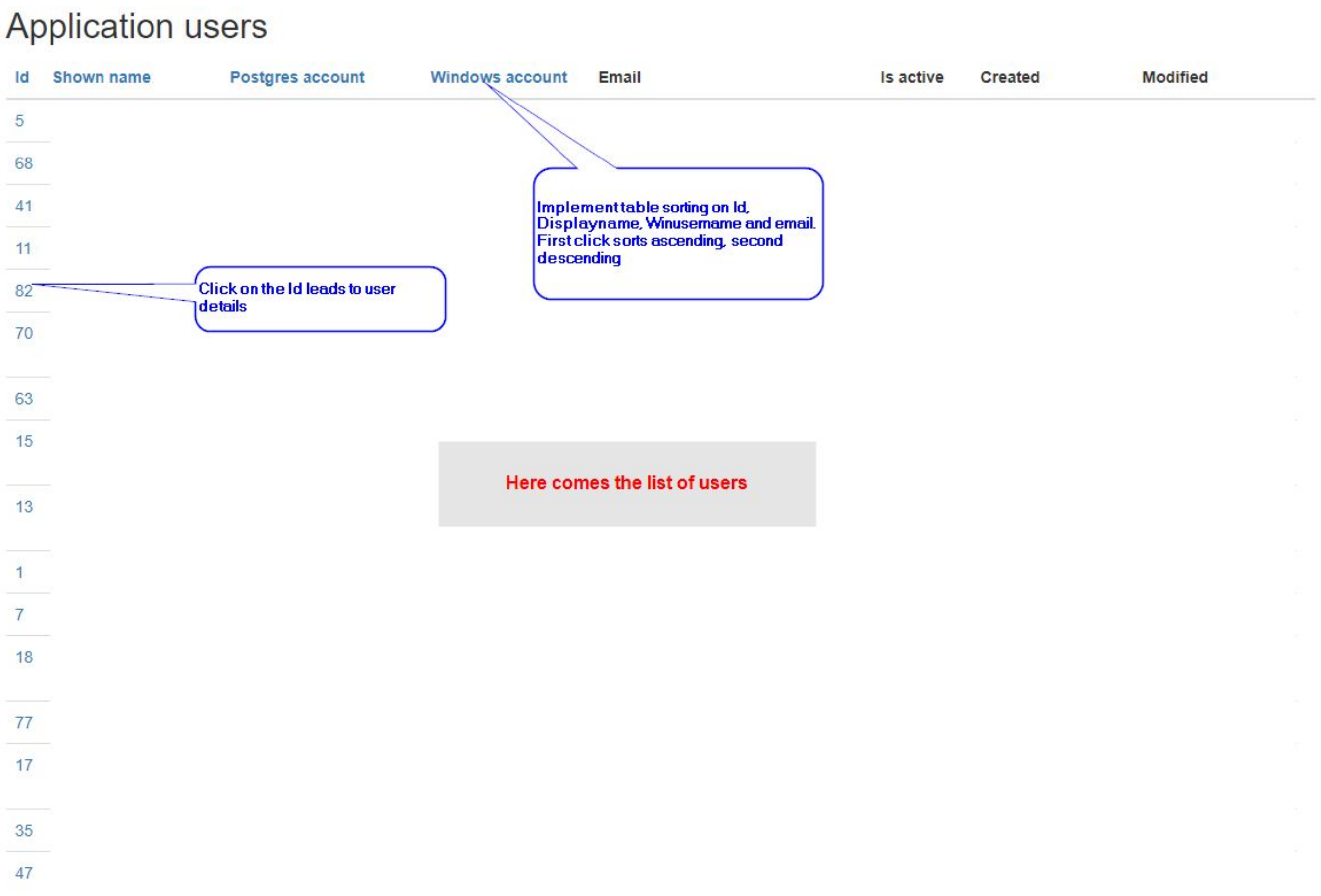


Title	IsVisible	Content	AvgRating
ARM Cortex: How to configure general CPU settings?	✓	Your CPU architecture is ARM Cortex and you would like to configure minimum architecture specific settings required to connect iSYSTEM BlueBox to the target microcontroller and perform CPU Reset Debug command, which establishes initial debug connection and gains user a complete control over the target microcontroller. **Configuration steps:** **CPU selection**: Correct selection of the target microcontroller (or SoC – System On Chip) is crucial for establishing the initial debug session. 1. Click the Hardware menu/Emulation Options/CPU tab to open the Emulation Options dialog. 2. Select ARM under the Family setting. Cortex (JTAG, SWD) is selected by default under the POD/iCARD/iTAG setting. 3. Next try to find and select a matching CPU family. 4. If your CPU family is not directly listed, select according Cortex architecture instead or 'SoC' for complex SoCs and then find your CPU under the CPU variant. If you fail to allocate your target device, try using a cross reference table available in Online help, listing all supported Cortex families.	

c. Commit and push changes to your branch (candidate-xx)



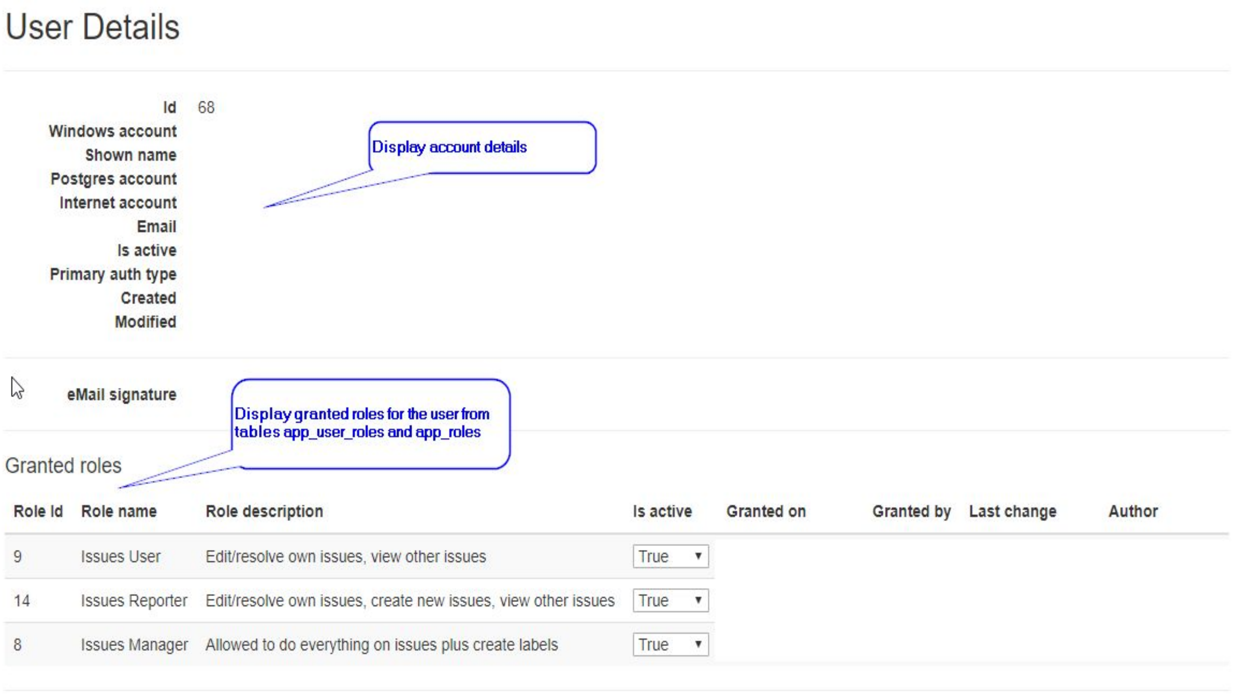
5. **Create a view, which lists all application users from the table ident.app_users**
   a. Page URL should be: /Account/Index


Application users

Id	Shown name	Postgres account	Windows account	Email	Is active	Created	Modified
5							
68							
41							
11							
82							
70							
63							
15							
13							
1							
7							
18							
77							
17							
35							
47							

Implement table sorting on Id, Displayname, Winusername and email. First click sorts ascending, second descending
Click on the Id leads to user details
Here comes the list of users

6. **Create a view, which shows selected user details**
   a. Page URL should be: /Account/Details/{Id}



User Details

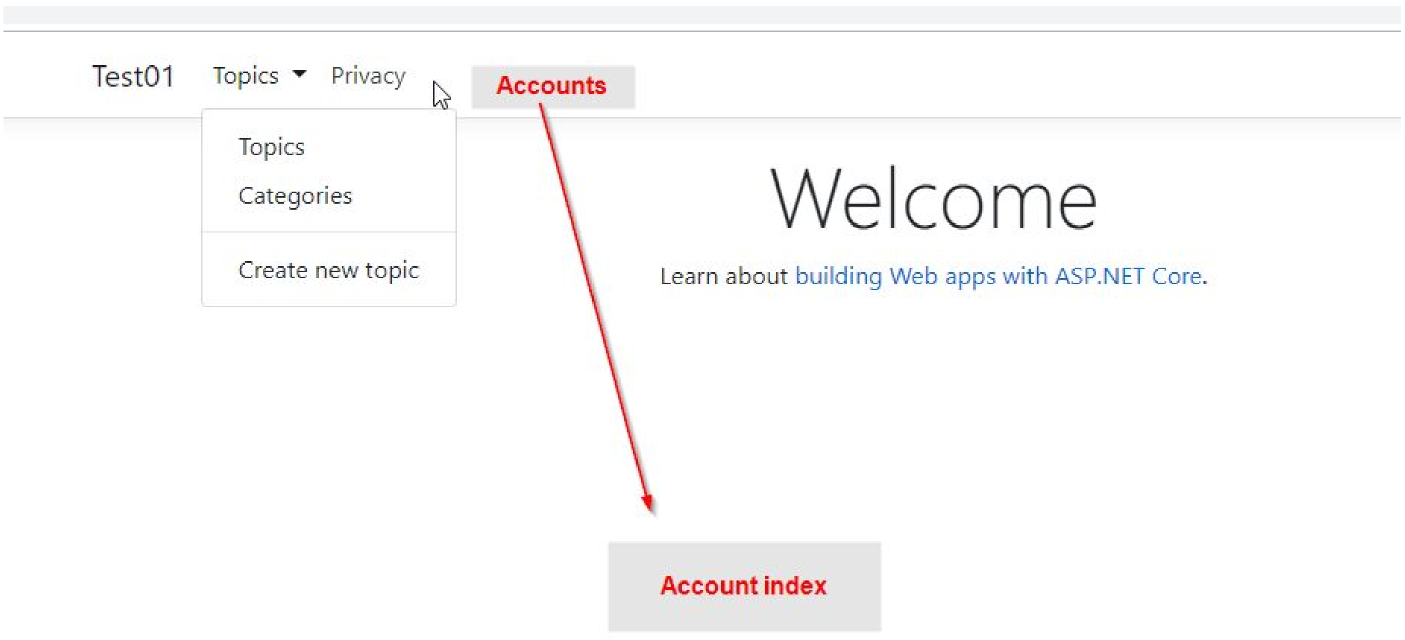
Id	68
Windows account	
Shown name	
Postgres account	
Internet account	
Email	
Is active	
Primary auth type	
Created	
Modified	
eMail signature	

Display account details
Display granted roles for the user from tables app_user_roles and app_roles

Granted roles

Role Id	Role name	Role description	Is active	Granted on	Granted by	Last change	Author
9	Issues User	Edit/resolve own issues, view other issues	True				
14	Issues Reporter	Edit/resolve own issues, create new issues, view other issues	True				
8	Issues Manager	Allowed to do everything on issues plus create labels	True				

7. **Add additional dropdown link in the menu, which leads to Account Index**



Welcome

Learn about building Web apps with ASP.NET Core.

8. **Push your changes to the repository**

Exam steps - part two

This part will be lead in the company premises.

Areas covered: AJAX, JSON, jQuery

1. Synchronize your repository branch on your local computer (Student-x)

2. Perform local Postgres DB synchronization from part one

3. Prepare search view for table topics.
   Search view should have a single field, which reacts on typing the keyword. It should display matching records on-the fly without reloading the page.

   Hints:
   1. Create RestApiController controller class, implement method “SearchTopics”. This method should return JSON object, containing list of topics.

   2. Fire AJAX / REST API call from the client to the URL ~/RestApi/SearchTopics/search=”search string”

   3. With jQuery, modify client HTML DOM and show results