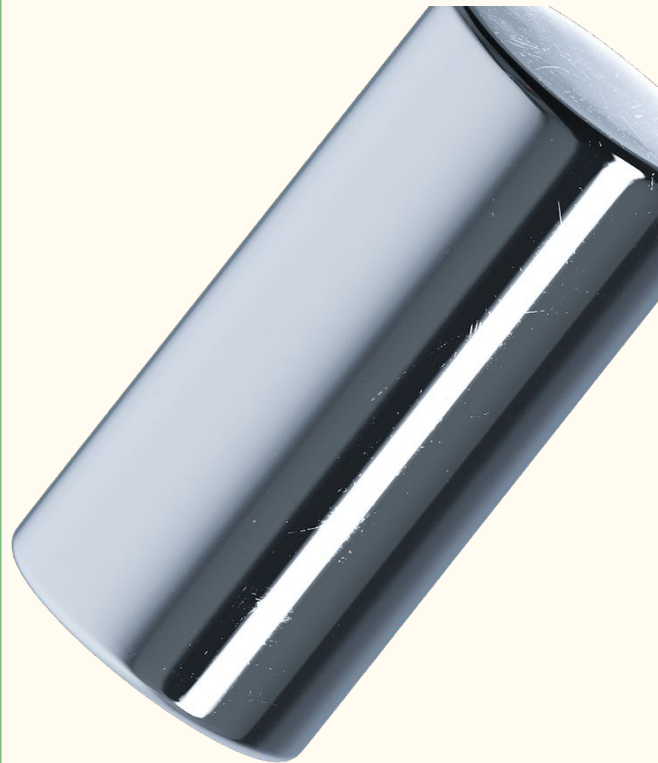


Installazione di un Server Dinamico con MySQL e Python

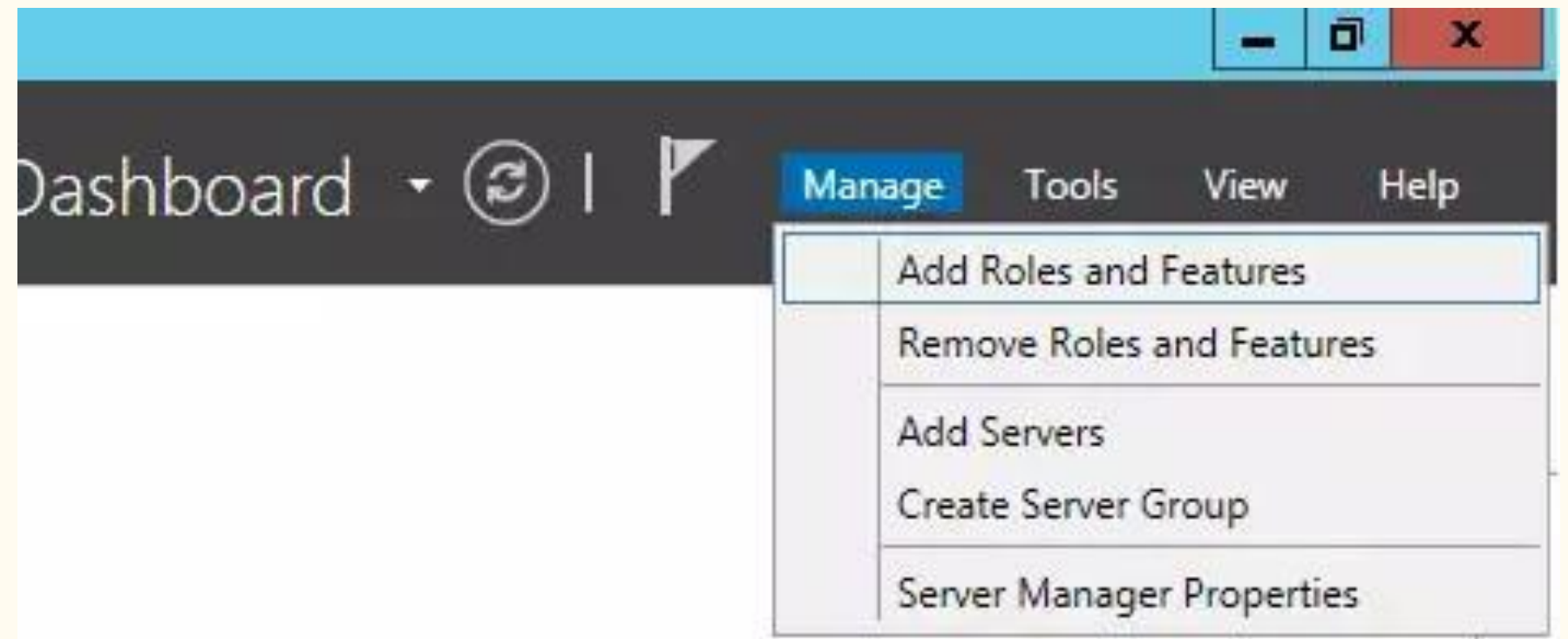


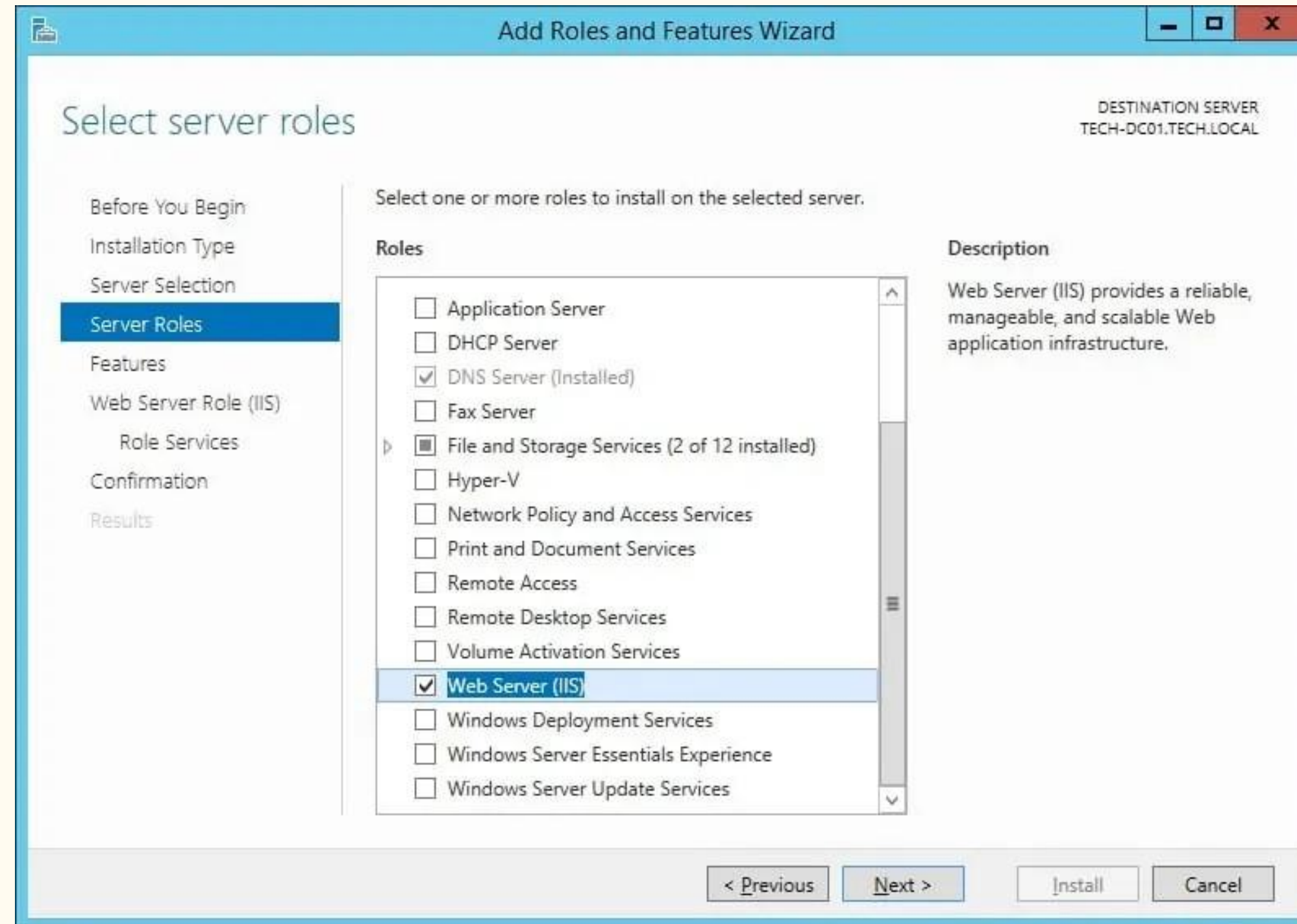


Configurazione Python Server Web

Installazione di IIS:

Aprire l'applicazione Server Manager.
Accedere al menu Gestisci e fare clic su
Aggiungi ruoli e funzionalità.



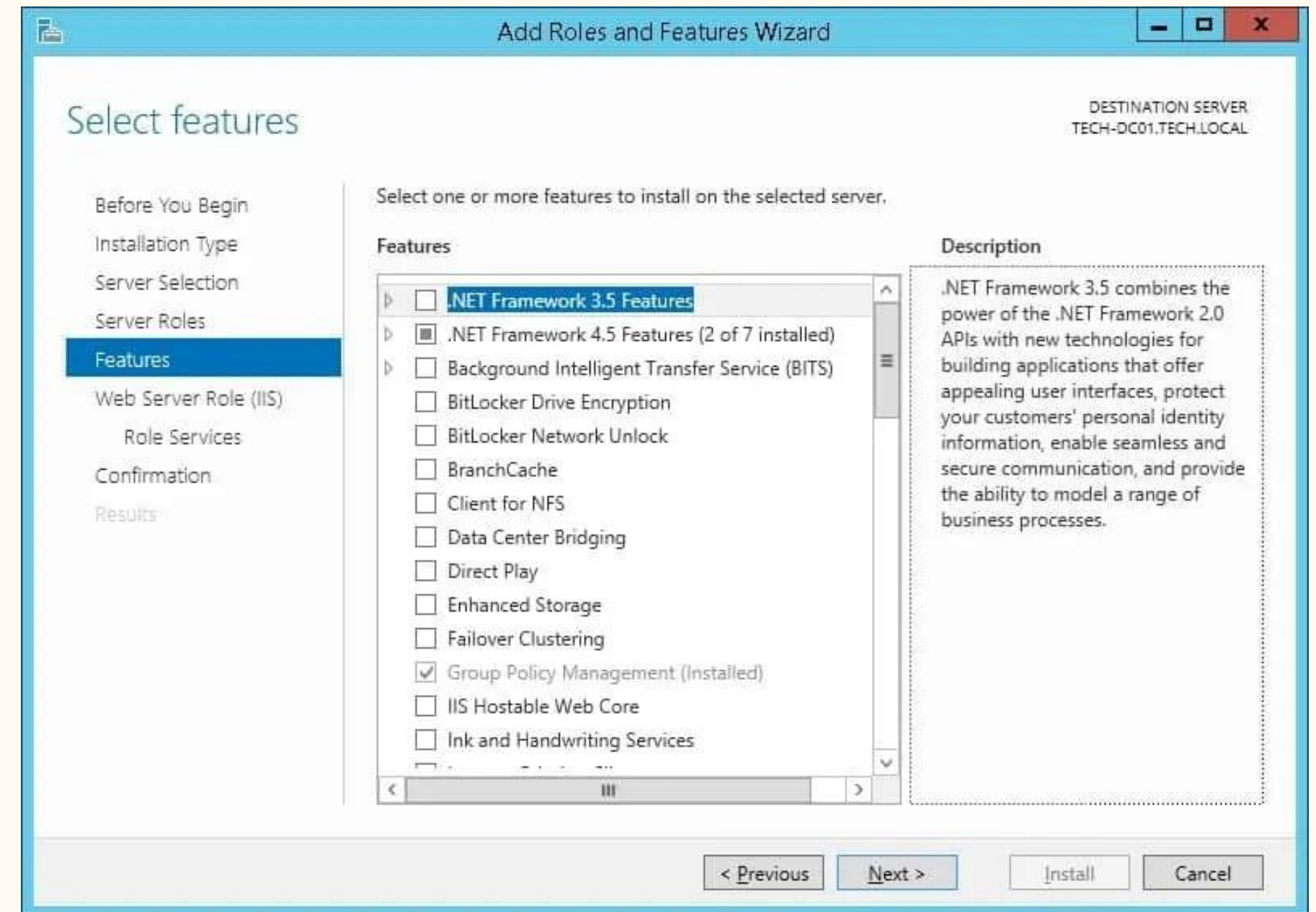


Nella schermata Ruoli server selezionare l'opzione denominata: IIS Web server Web. Fare clic sul pulsante Avanti.

Nella schermata
seguente, fare
clic sul pulsante
Aggiungi
funzionalità.



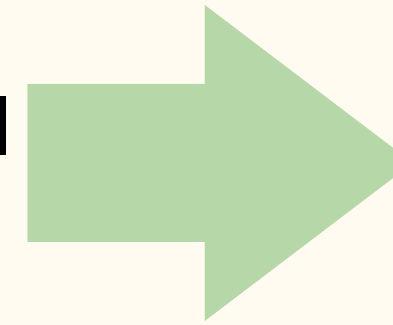
Nella schermata
Funzioni, fare clic sul
pulsante Avanti.



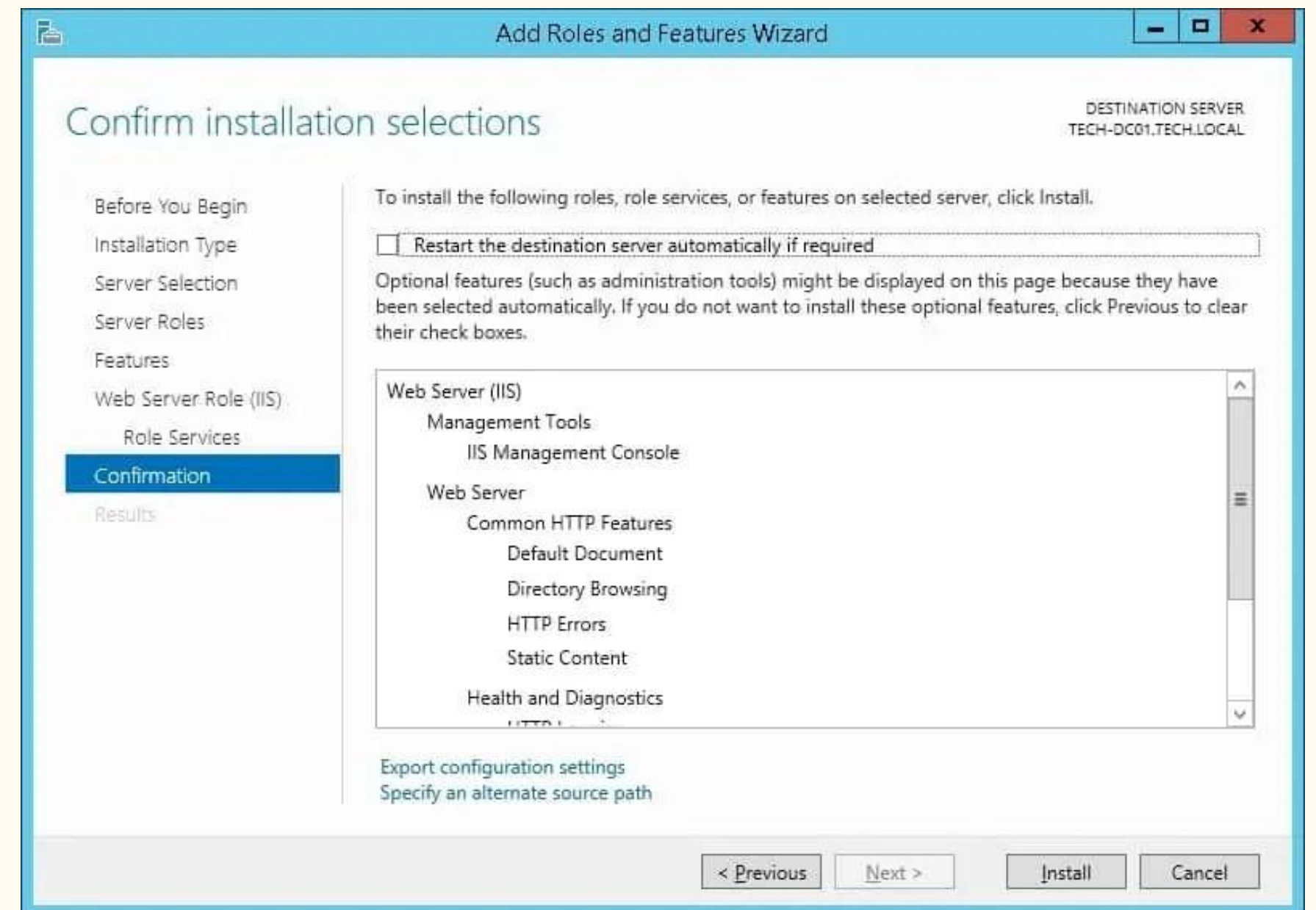
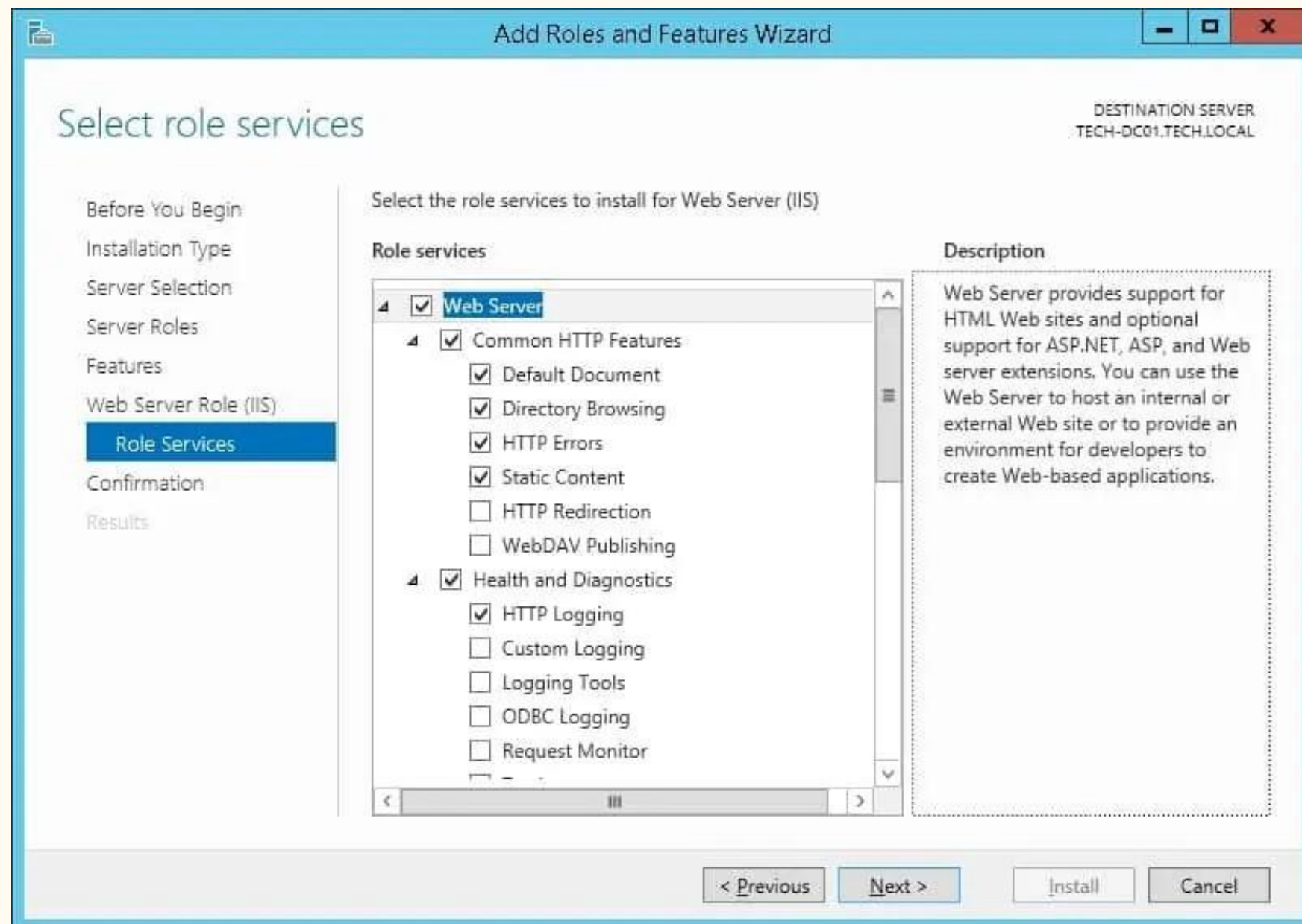
Nella schermata
Servizio ruolo fare clic
sul pulsante Avanti.



Nella schermata
Riepilogo, fare clic sul
pulsante Installa.

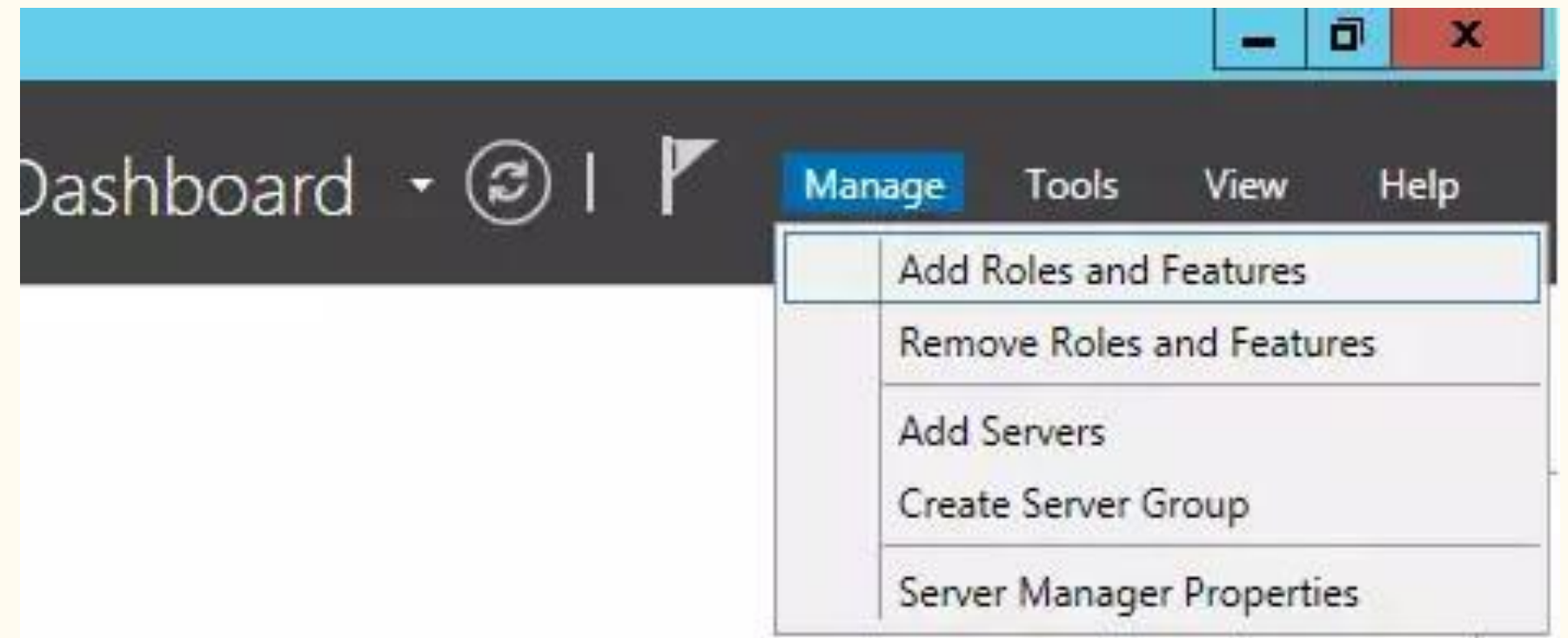


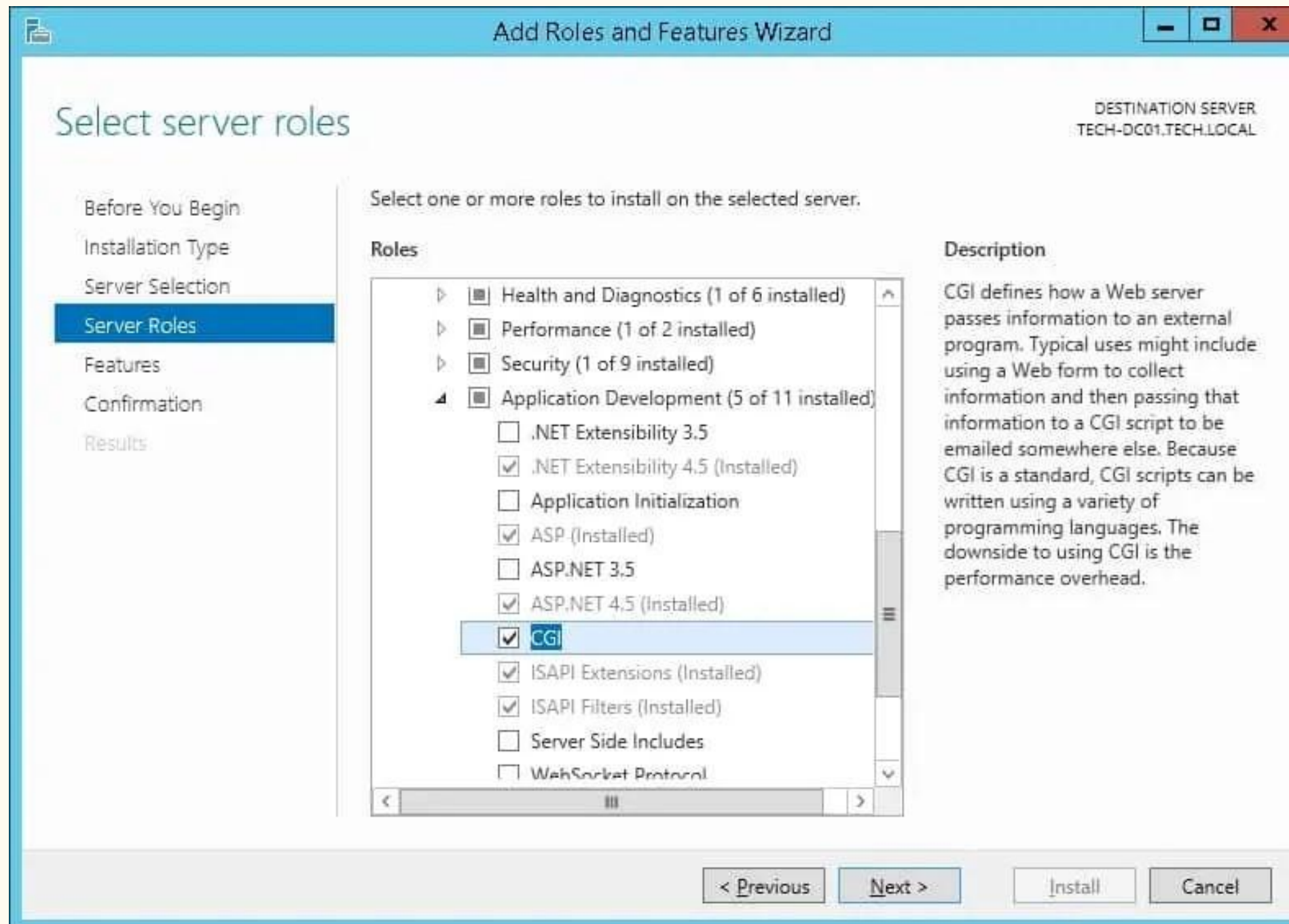
Congratulazioni!
L'installazione del servizio IIS
in un computer che esegue
Windows è stata completata.



Esercitazione IIS - Abilitazione della funzionalità CGI:

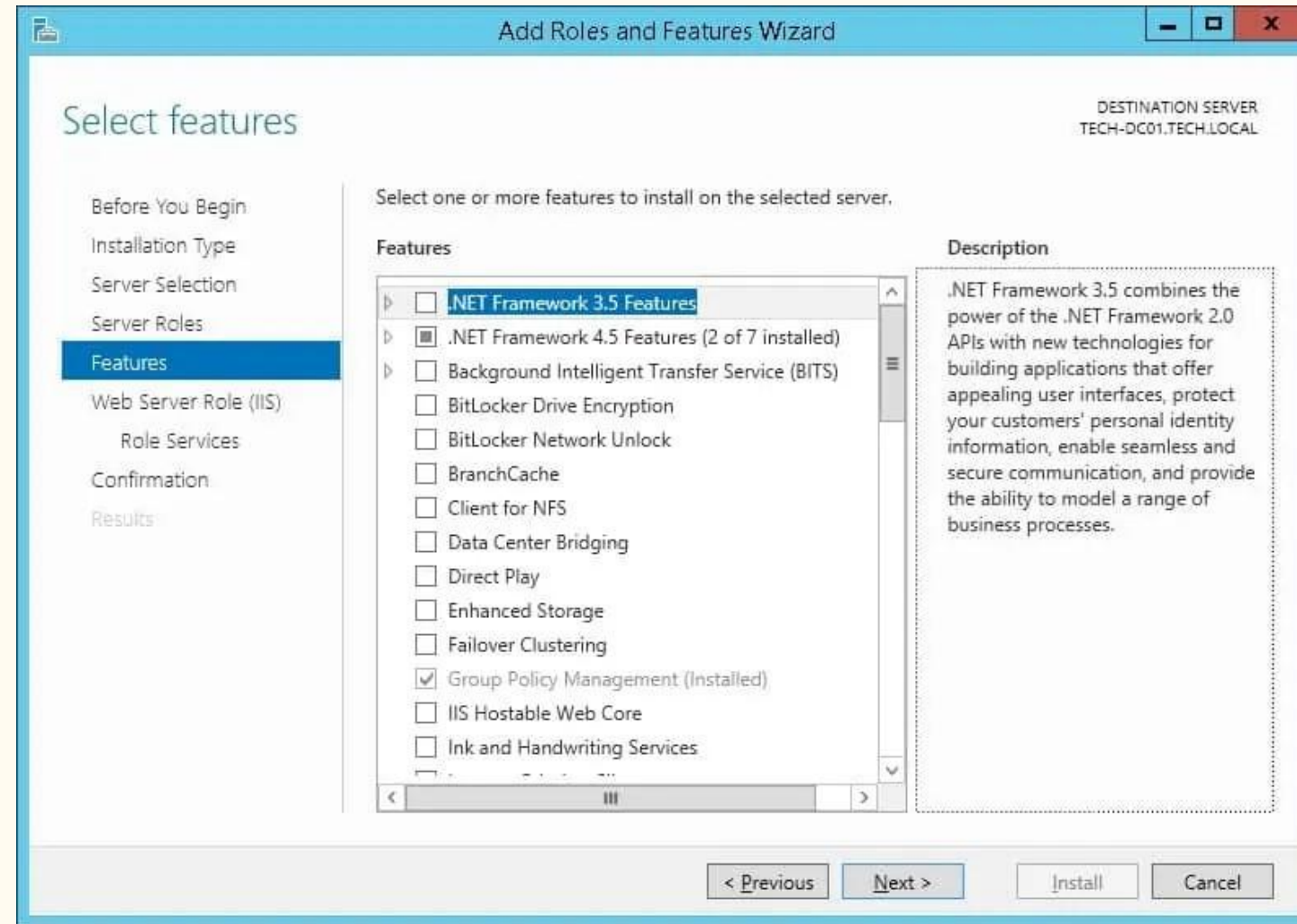
Aprire l'applicazione Server Manager.
Accedere al menu Gestisci e fare clic su
Aggiungi ruoli e funzionalità.





Nella schermata Ruoli server espandere la voce denominata: IIS del server Web.
Accedere al menu Sviluppo applicazioni e selezionare l'opzione denominata: CGI
Fare clic sul pulsante Avanti.

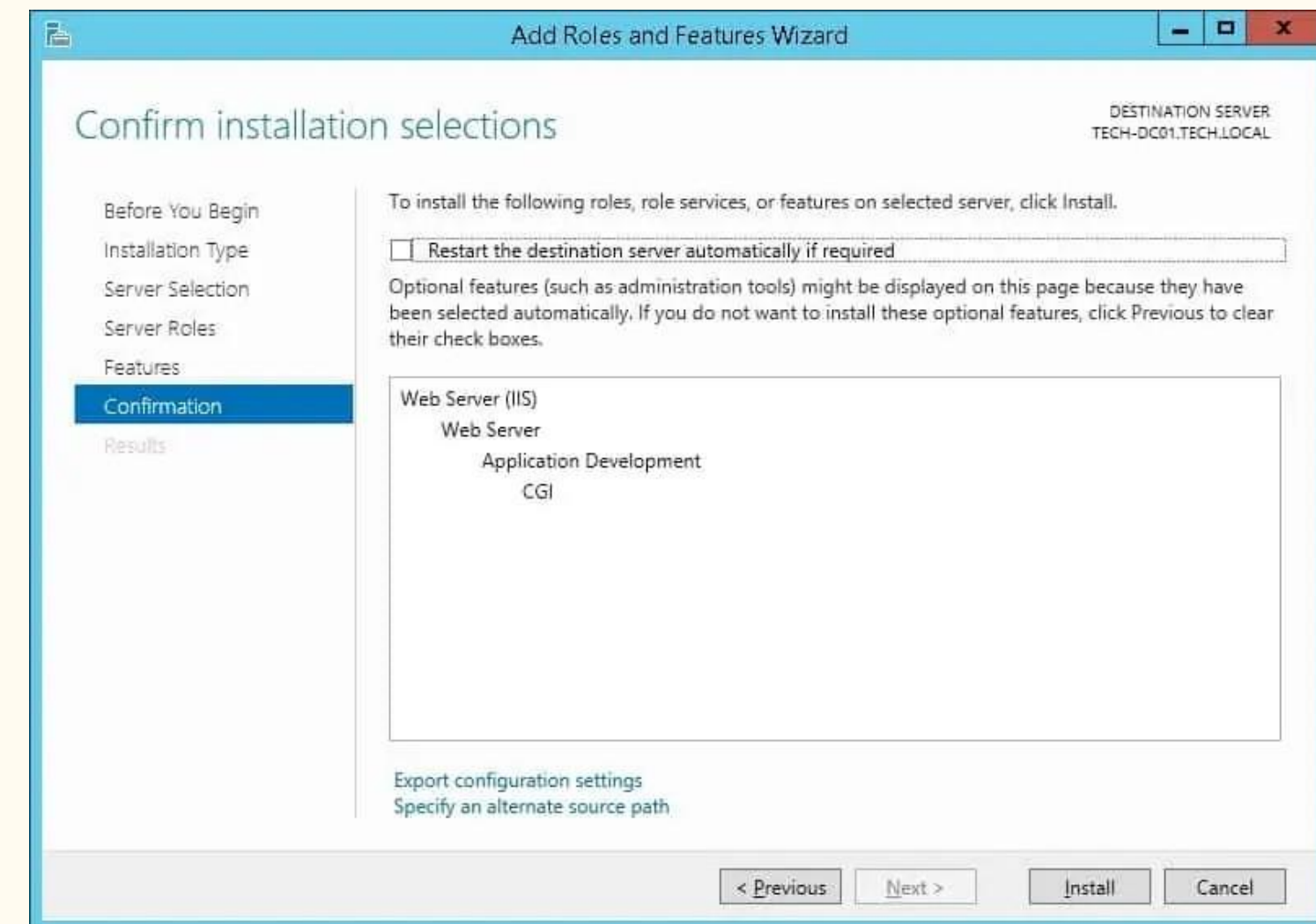
Nella schermata Funzioni, fare clic sul pulsante Avanti.



Congratulazioni! L'installazione della funzionalità CGI in IIS è stata completata.



Nella schermata Riepilogo, fare clic sul pulsante Installa.



Esercitazione Windows - Installazione Python:

Accedi al sito web di Python e scarica
l'ultima versione del programma di
installazione di Python.



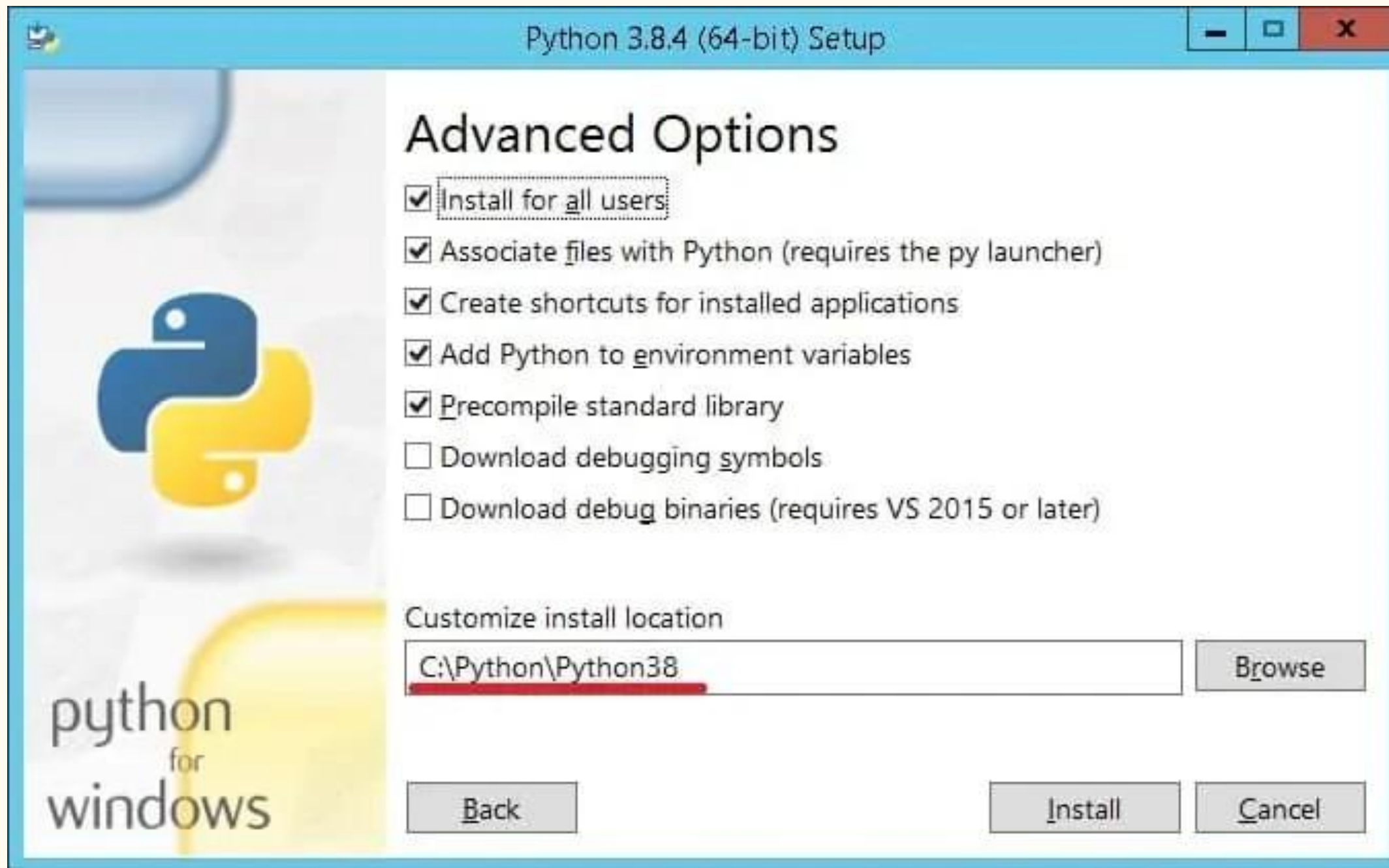
-Come amministratore,
avviare l'installazione di
Python.

-Selezionare entrambe le
caselle di controllo nella
parte inferiore dello
schermo.



Fare clic sul pulsante per personalizzare
l'installazione di Python.





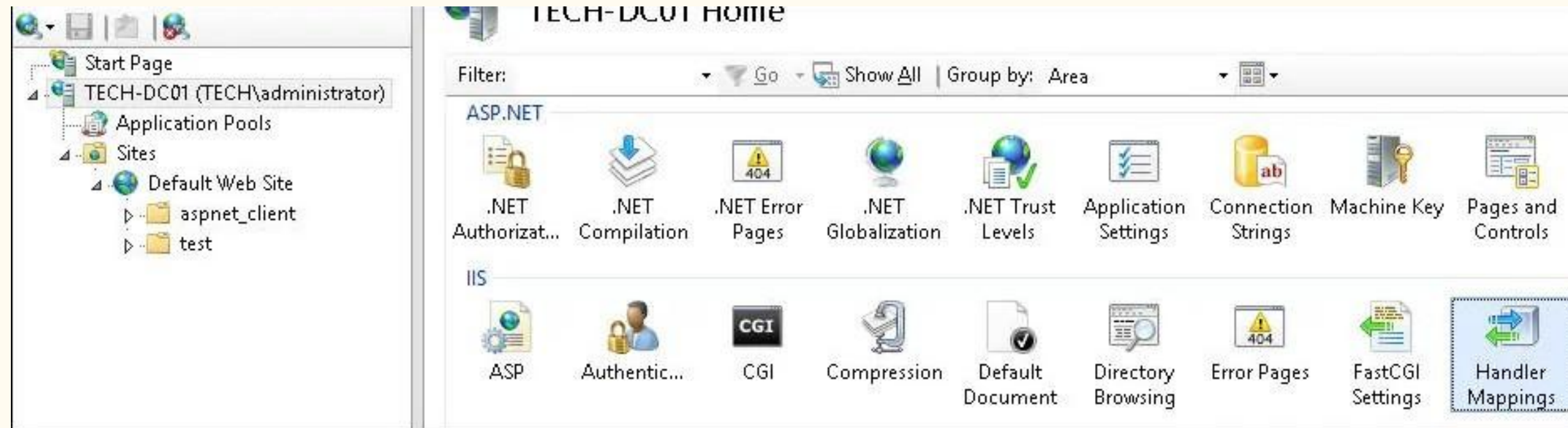
- Selezionare tutte le caselle di controllo e fare clic sul pulsante Avanti.
- Selezionare la casella di controllo denominata: Installa per tutti gli utenti.
- Modificare il percorso di installazione di Python nella radice dell'unità C.

Esercitazione IIS - Abilitare Python sul server IIS:

- Avviare l'applicazione denominata:
Gestione IIS.
- Nell'applicazione Gestione IIS,
selezionare il nome del server IIS.



Nella parte destra dello schermo, accedere all'opzione denominata: Mapping gestori

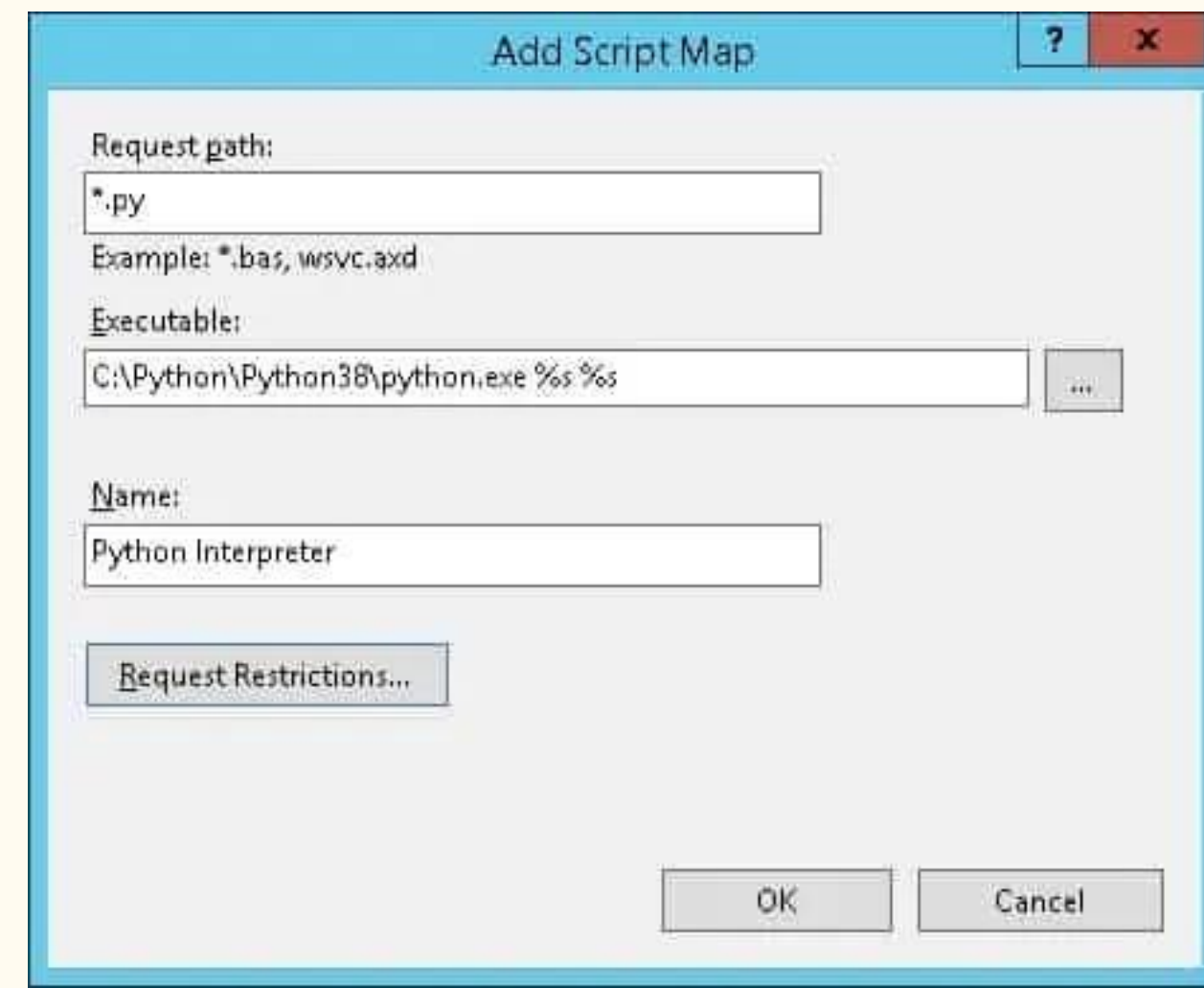


Selezionare l'opzione denominata: Aggiungi mappa script



Eseguire la seguente configurazione:

- Request Path - *.py
- Executable - C:\Python\Python312\python.exe %s %s
- Interpreter - Python Interpreter Fare clic sul pulsante OK.



-Se viene visualizzato il seguente messaggio, fare clic sul pulsante Sì.

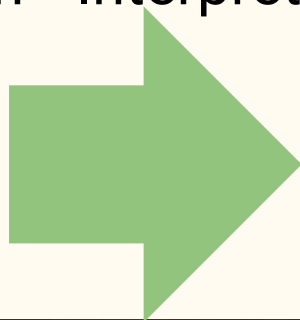
-Riavviare il servizio IIS.

-Python è stato abilitato correttamente sul server IIS.



Eseguire la seguente configurazione:

- Request Path - *.py
- Executable - C:\Python\Python312\python.exe %s %s
- Interpreter - Python Interpreter Fare clic sul pulsante OK.



Add Script Map

Request path:
*.py
Example: *.bas, wsvc.axd

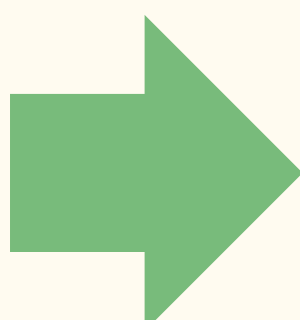
Executable:
C:\Python\Python38\python.exe %s %s

Name:
Python Interpreter

Request Restrictions...

OKCancel

Se viene visualizzato il seguente messaggio, fare clic sul pulsante Sì.



Add Script Map

Do you want to allow this ISAPI extension? Click "Yes" to add the extension with an "Allowed" entry to the ISAPI and CGI Restrictions list or to update an existing extension entry to "Allowed" in the ISAPI and CGI Restrictions list.

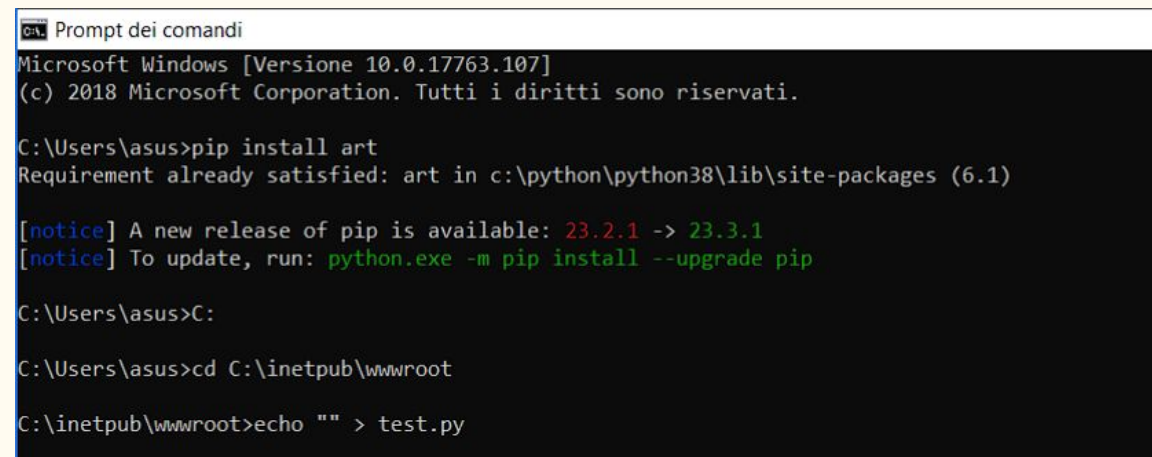
YesNoCancel

Riavviare il servizio IIS. Python è stato abilitato correttamente sul server IIS.

Verifica Python

1. Eseguire cmd come amministratore
2. Eseguire il comando:

pip install art



```
Prompt dei comandi
Microsoft Windows [Versione 10.0.17763.107]
(c) 2018 Microsoft Corporation. Tutti i diritti sono riservati.

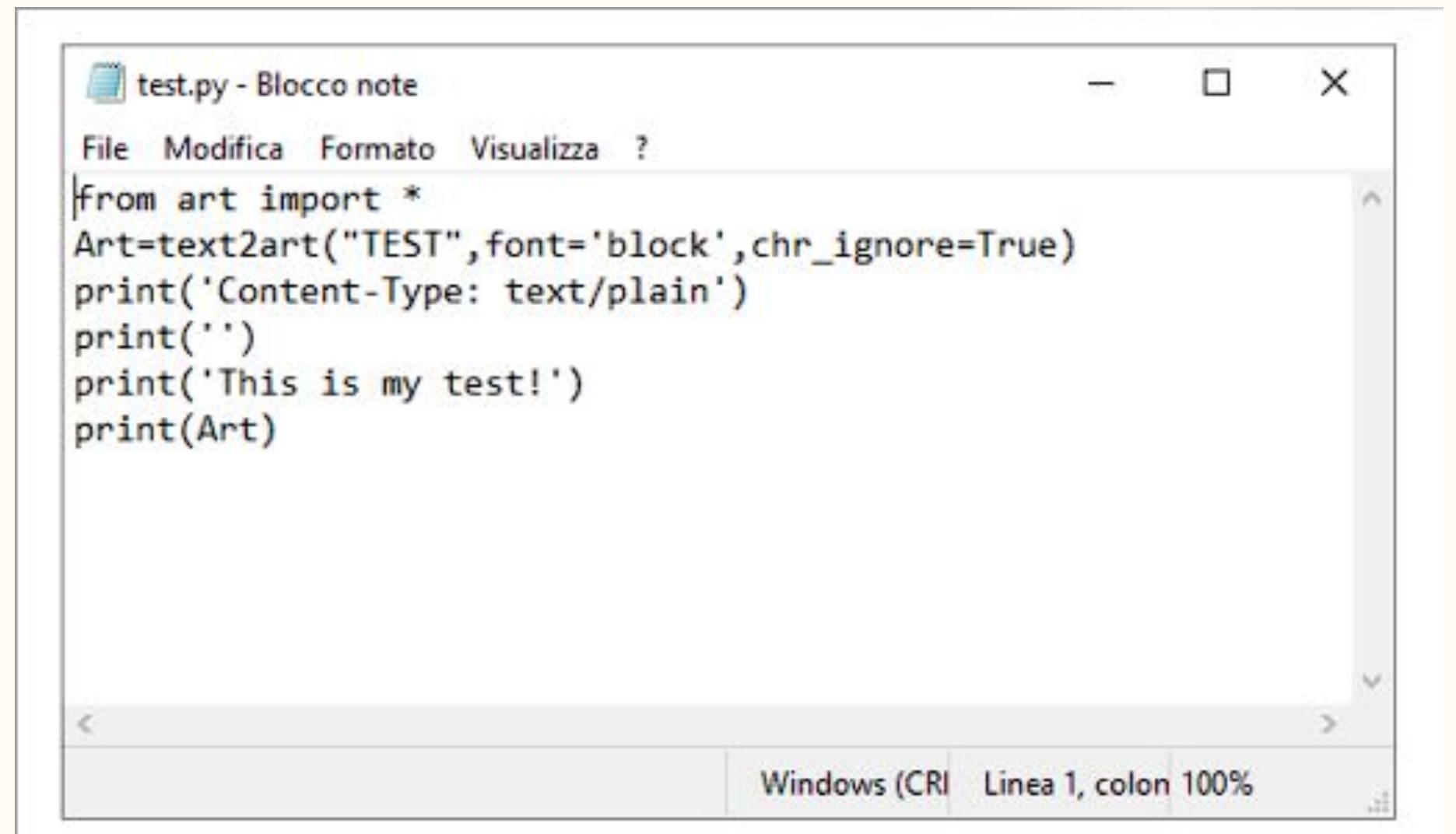
C:\Users\asus>pip install art
Requirement already satisfied: art in c:\python\python38\lib\site-packages (6.1)

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\asus>C:
C:\Users\asus>cd C:\inetpub\wwwroot
C:\inetpub\wwwroot>echo "" > test.py
```

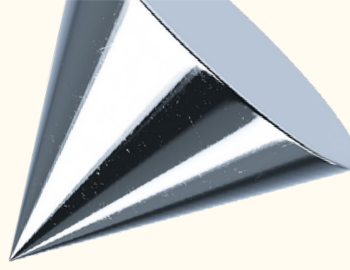
3. Creare un file.py con blocco note e mettere all'interno:

```
from art import *
Art=text2art("TEST",font='block',chr_ignore=True)
print('Content-Type: text/plain')
print("")
print('This is my test!')
print(Art)
```

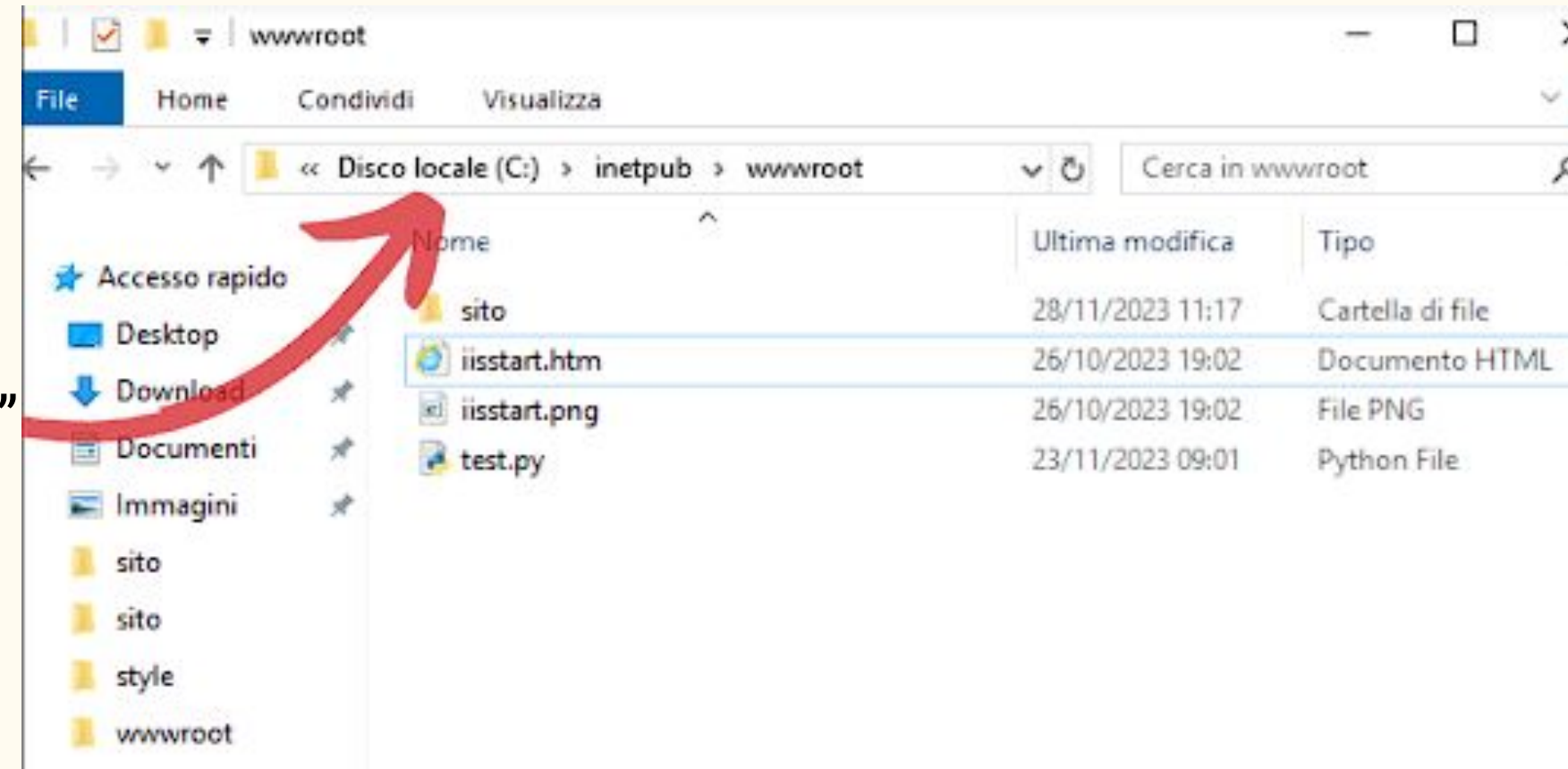


```
test.py - Blocco note
File  Modifica  Formato  Visualizza  ?
from art import *
Art=text2art("TEST",font='block',chr_ignore=True)
print('Content-Type: text/plain')
print('')
print('This is my test!')
print(Art)
```

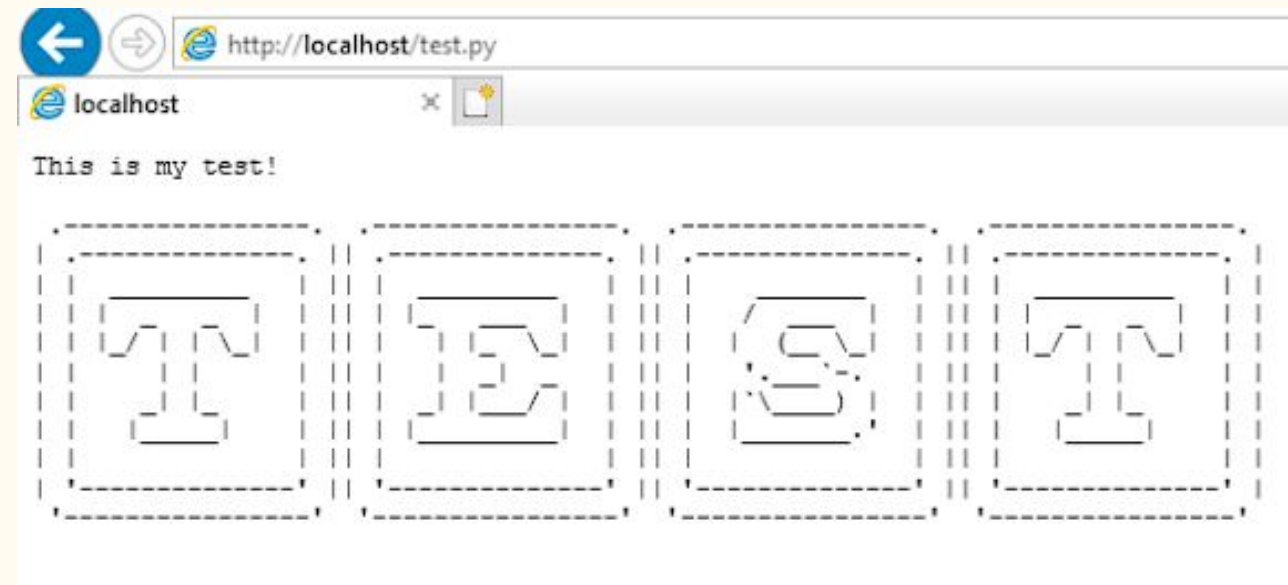

Verifica Python



Spostare il file nella cartella “wwwroot”



Per eseguirlo: immettere l'indirizzo indicato, nella barra di ricerca del browser





Come installare
MySQL in Windows 10



Download del software necessario:

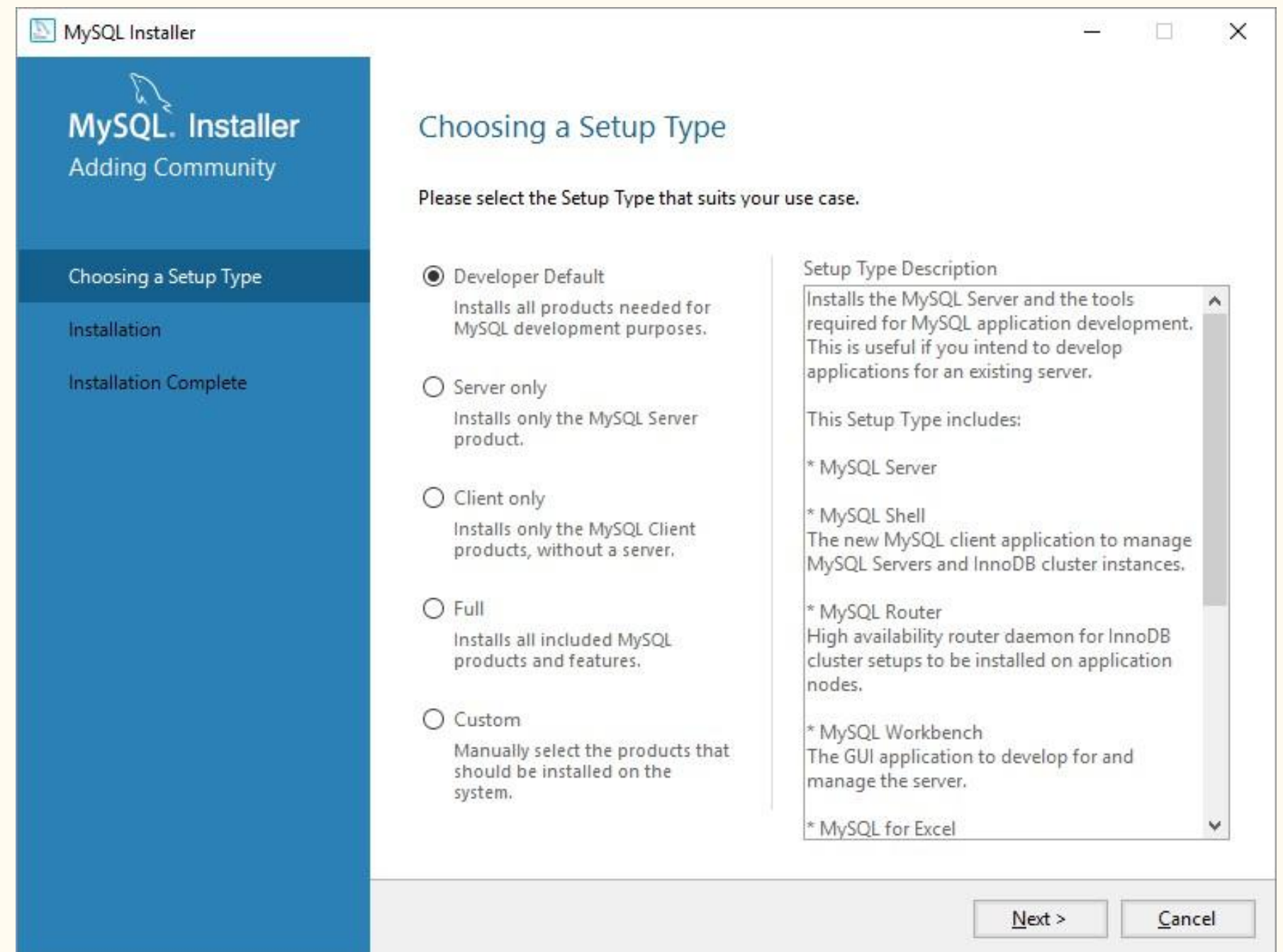
The screenshot displays the MySQL Community Server 5.7.21 download page. The navigation bar at the top includes links for Enterprise, Community, Yum Repository, APT Repository, SUSE Repository, Windows, and Archives. The main content area is titled 'MySQL Community Server 5.7.21' and includes tabs for 'Generally Available (GA) Releases' and 'Development Releases'. Below the title, there are dropdown menus for 'Select Operating System' (set to 'Microsoft Windows') and 'Select OS Version' (set to 'All'). A 'Looking for previous GA versions?' link is also present. The 'Recommended Download' section features a large graphic for the 'MySQL Installer for Windows' with the text 'All MySQL Products. For All Windows Platforms. In One Package.' and a 'Go to Download Page >' button. Below this, the 'Other Downloads' section contains a table with the following data:

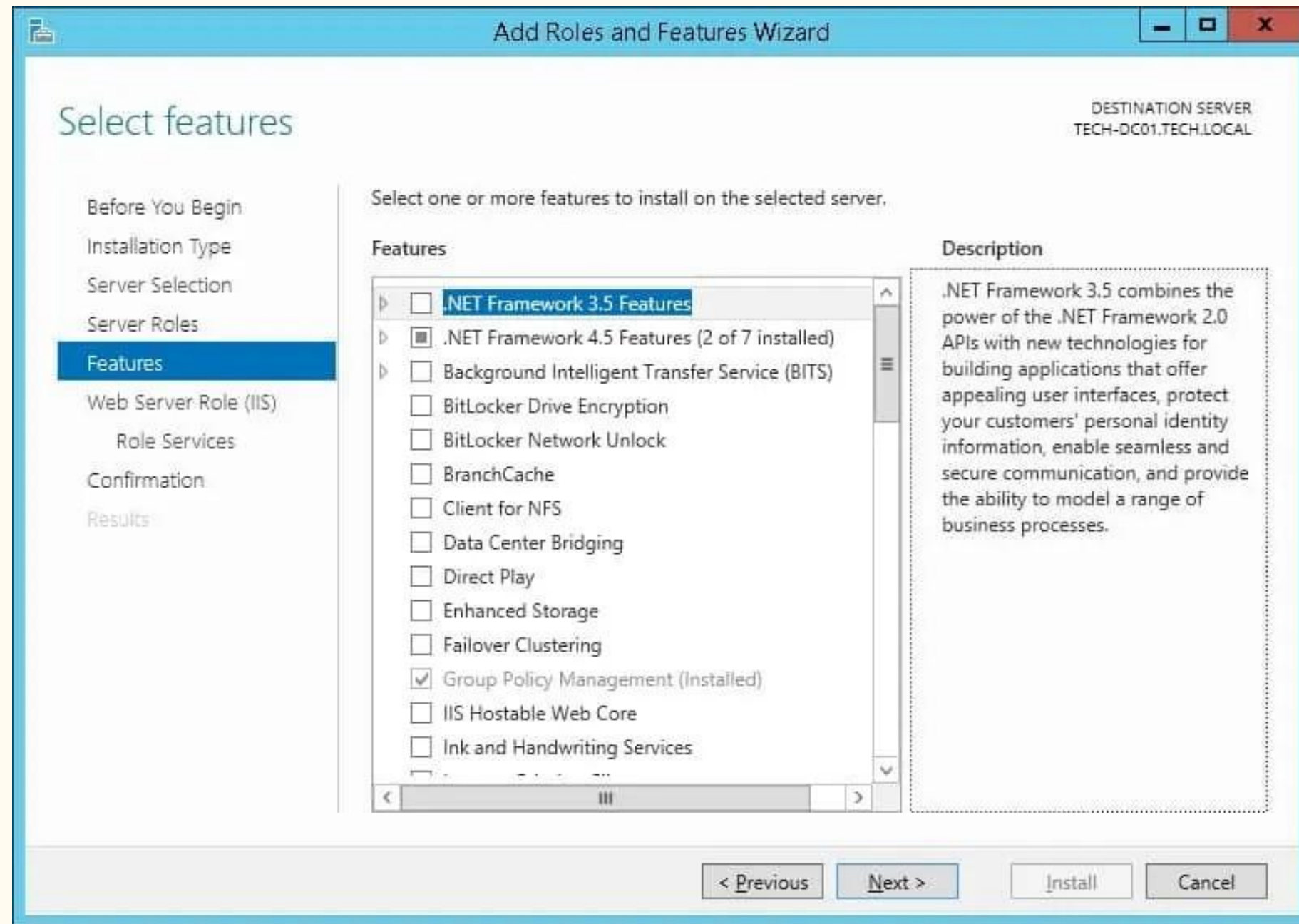
Download Link	Version	Size	Action
Windows (x86, 32-bit), ZIP Archive (mysql-5.7.21-win32.zip)	5.7.21	306.7M MD5: fc7cfdeab93c51ad85aaf92b77424dcd Signature	Download
Windows (x86, 64-bit), ZIP Archive (mysql-5.7.21-winx64.zip)	5.7.21	319.2M MD5: ffa7c3f37da083c810072935cf6b9cb6 Signature	Download
Windows (x86, 32-bit), ZIP Archive Debug Binaries & Test Suite (mysql-5.7.21-win32-debug-test.zip)	5.7.21	372.7M MD5: 81a100ce684189e37ae4b3f365f50eb3 Signature	Download
Windows (x86, 64-bit), ZIP Archive	5.7.21	382.6M	Download

-Per scaricare il pacchetto di installazione, puntare la pagina del browser all'indirizzo: <https://dev.mysql.com/downloads/mysql/>.

Installazione:

- Terminato il download, potrai finalmente avviare l'installazione del pacchetto eseguendo il file EXE appena scaricato.
- Appena avviato, ti verrà presentata questo pannello, tramite il quale potrai verificare il tipo di installazione desiderata.
- Installare la versione "Full"





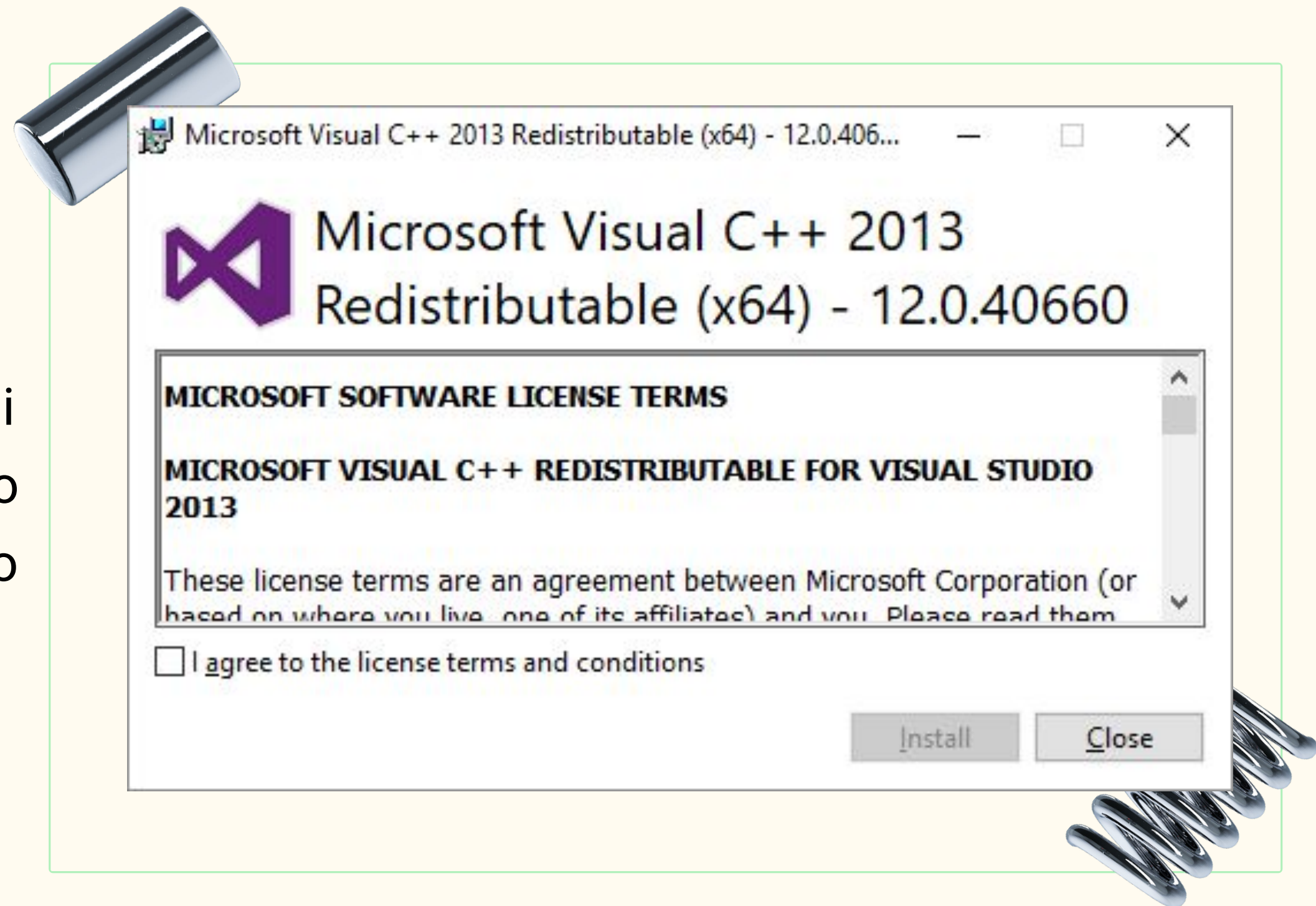
La prossima schermata che ti verrà mostrata, serve all'installatore (il pacchetto exe che hai appena avviato) per verificare che il tuo sistema abbia tutto ciò che gli serve per fare un'installazione coerente.

-Pertanto qui potrai cliccare su "Execute" per fare in modo che vengano risolte tutte le dipendenze per l'installazione.

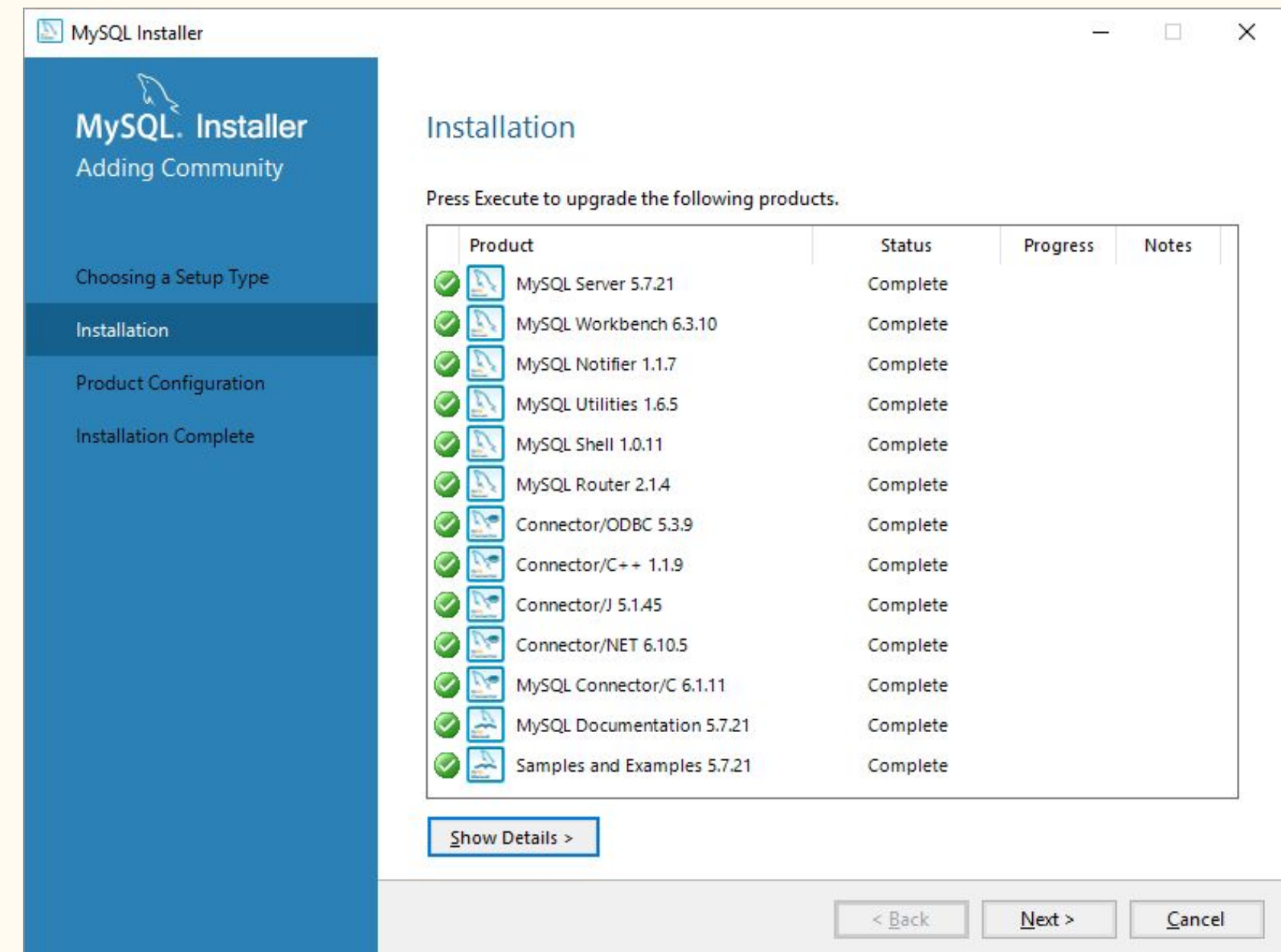
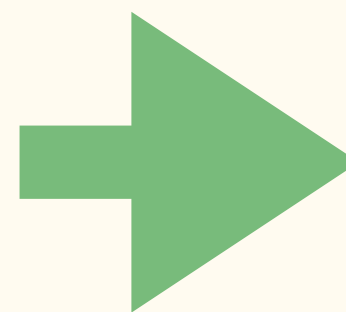
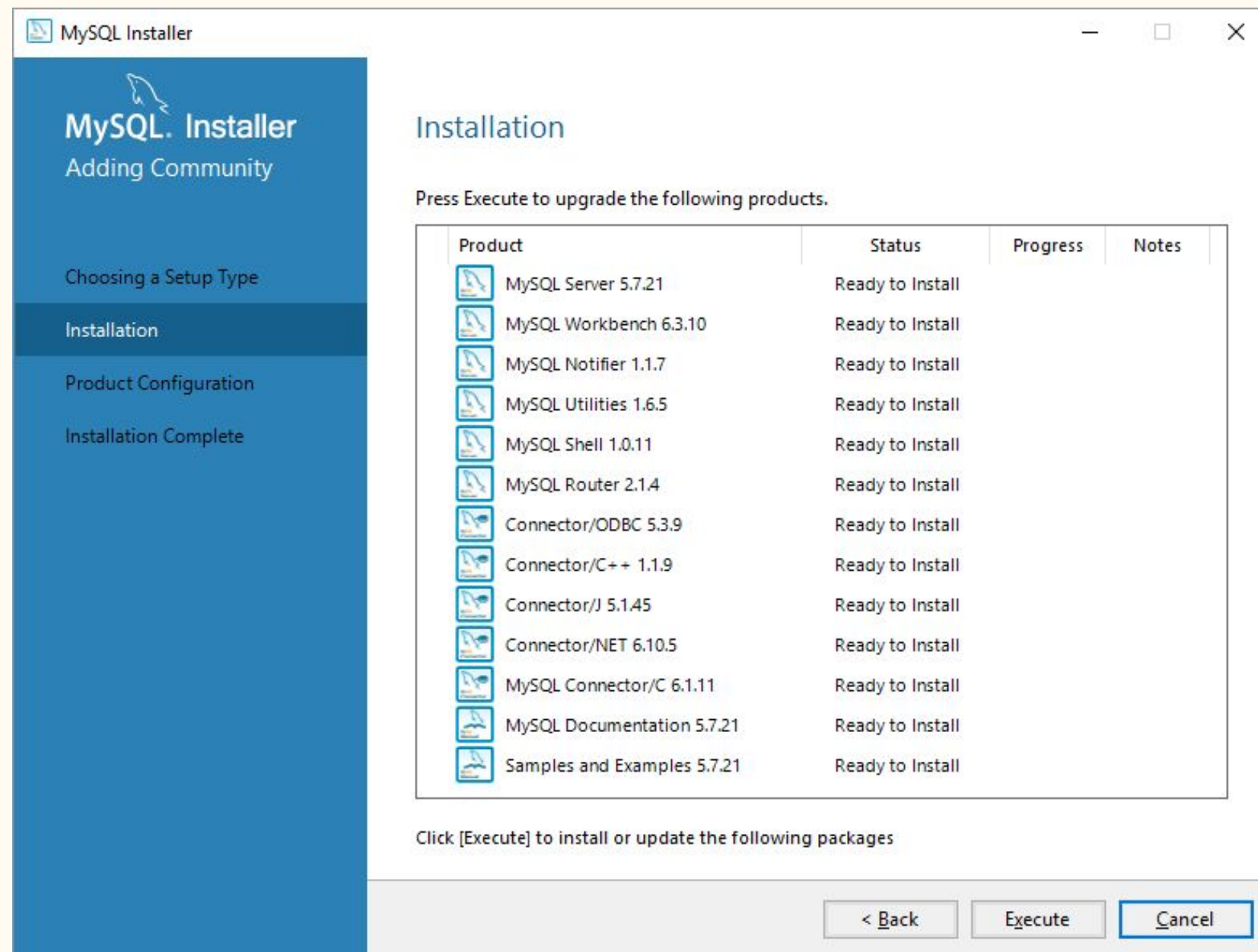
-Durante questa fase, che può prendere alcuni minuti, verranno scaricati tutti i pacchetti necessari per risolvere le dipendenze.

-Una delle richieste di conferma per l'installazione automatica delle dipendenze.

-Tuttavia, potrebbero essere necessari diversi passi simili a questo, che dipendono fondamentalmente da cosa viene rilevato mancante nel tuo sistema Windows.



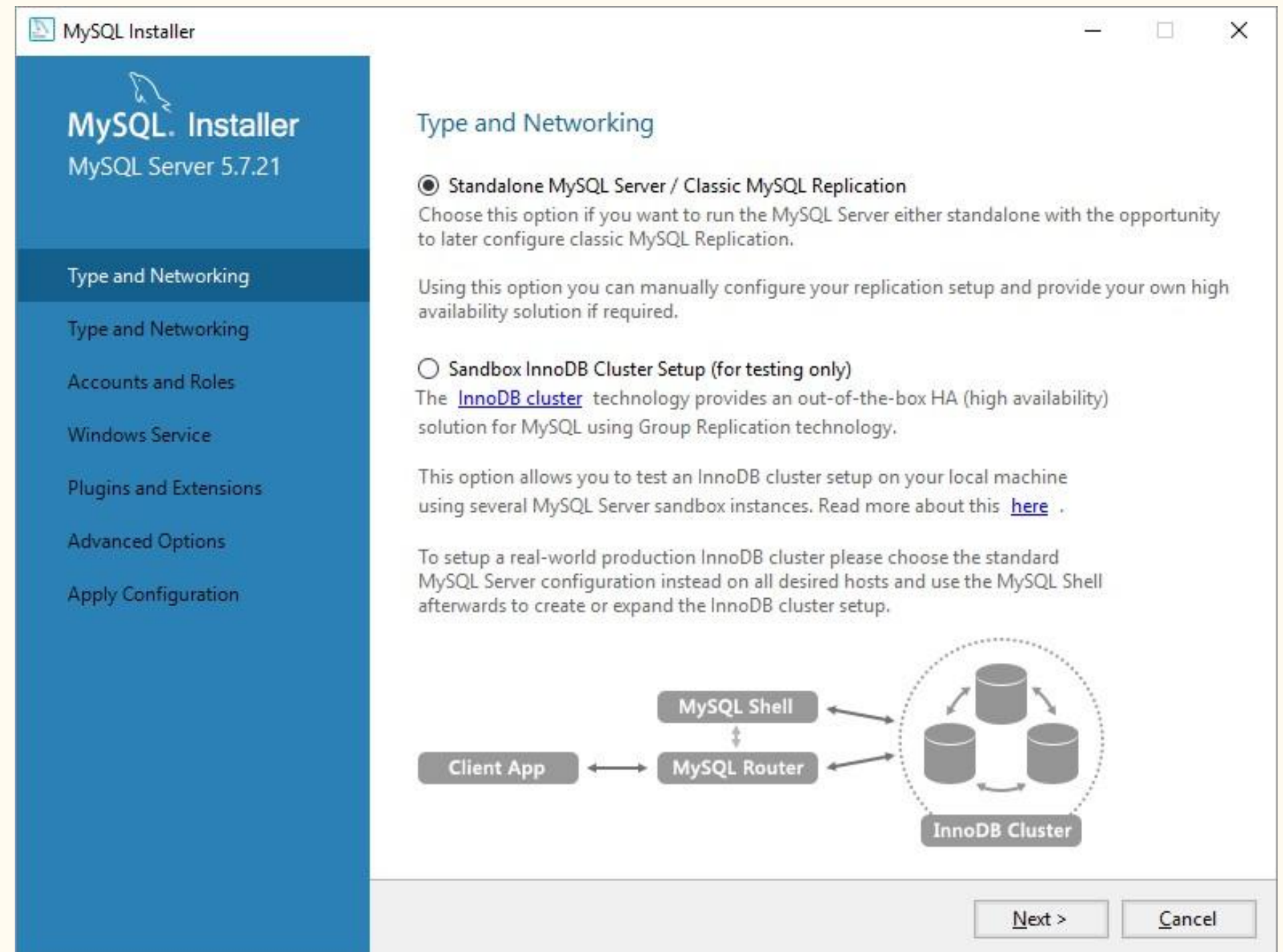
- Per avviare, premere Execute.
- Quando questa fase sarà terminata ti verrà mostrato un pannello simile al precedente, che ti indicherà se qualcosa è andato storto oppure no.
- Premi Next per completare questa fase dell'installazione.



Configurazione:

Una volta completata questa fase, l'installatore non terminerà, ma provvederà a guidarti nella fase successiva per la completa configurazione dei pacchetti appena installati, mostrandoti questa schermata iniziale.

-Premendo "Next" potrai proseguire con tutte le configurazioni necessarie.



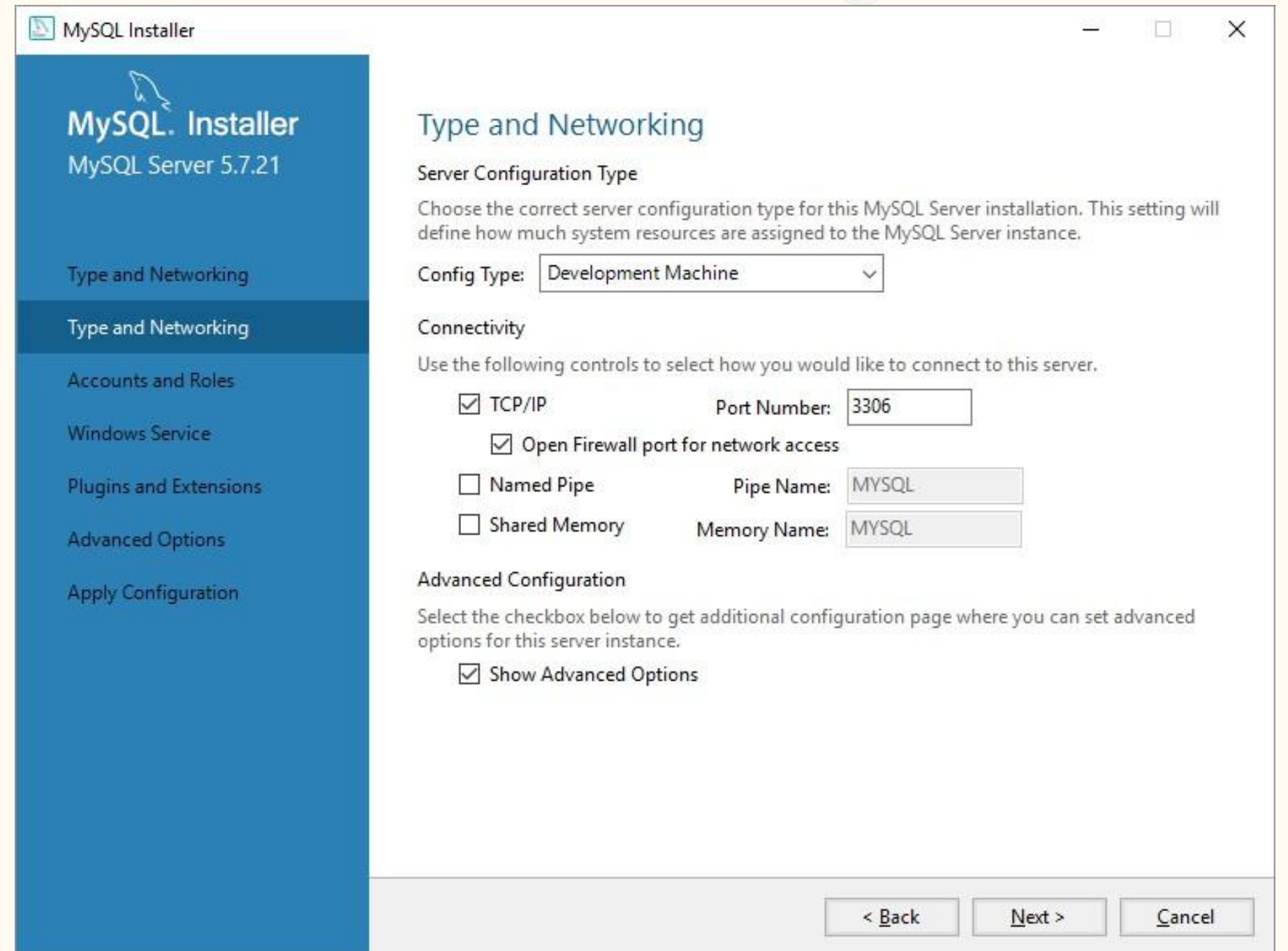
Tipo di Server e Rete:

.Qui potrai configurare sia il tipo di server che soprattutto la rete.

-Tipo di configurazione, scelta per il nostro Mysql:

- Development Machine – Sistema di sviluppo con molte applicazioni accessorie.

-Nello stesso pannello di configurazione, dovrai specificare il tipo di connessione che vuoi rendere disponibile per poter accedere a MySQL, noi non abbiamo modificato nulla



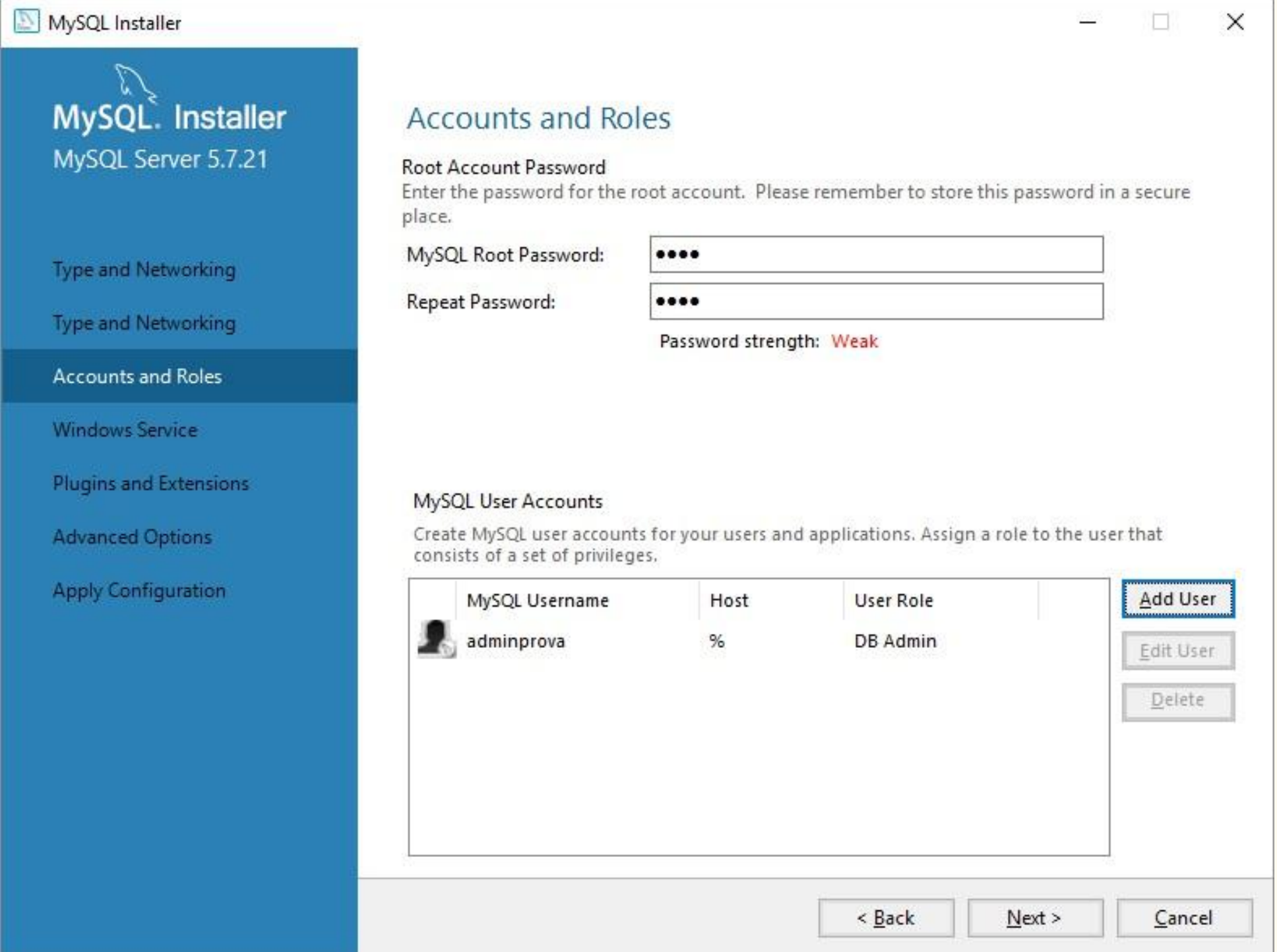
The screenshot shows the 'MySQL Installer' window for 'MySQL Server 5.7.21'. The left sidebar contains a list of configuration steps: 'Type and Networking' (selected), 'Accounts and Roles', 'Windows Service', 'Plugins and Extensions', 'Advanced Options', and 'Apply Configuration'. The main area is titled 'Type and Networking' and contains the following sections:

- Server Configuration Type:** A text box with the instruction: 'Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.' Below it, a dropdown menu labeled 'Config Type:' is set to 'Development Machine'.
- Connectivity:** A text box with the instruction: 'Use the following controls to select how you would like to connect to this server.' Below it, there are four options:
 - ☒ TCP/IP: Port Number: 3306
 - ☒ Open Firewall port for network access
 - ☐ Named Pipe: Pipe Name: MYSQL
 - ☐ Shared Memory: Memory Name: MYSQL
- Advanced Configuration:** A text box with the instruction: 'Select the checkbox below to get additional configuration page where you can set advanced options for this server instance.' Below it, there is a checkbox labeled 'Show Advanced Options' which is checked.

At the bottom right, there are three buttons: '< Back', 'Next >', and 'Cancel'.

Account e Ruoli:

- Per poter accedere a MySQL bisogna assolutamente creare account e ruoli specifici.
- In questo passo, puoi definire gli account ed i ruoli per l'accesso a MySQL.
- La prima cosa da fare in assoluto è definire la Root password, nella parte alta del pannello.
- Mentre potrai sfruttare la parte inferiore del pannello per creare tutti gli account che desideri, cliccando su Add User e specificando le credenziali per il nuovo account che stai creando.



The screenshot shows the 'MySQL Installer' window for 'MySQL Server 5.7.21'. The left sidebar has a blue background with the MySQL logo and the text 'MySQL. Installer MySQL Server 5.7.21'. Below this, there are five menu items: 'Type and Networking', 'Type and Networking', 'Accounts and Roles' (which is highlighted), 'Windows Service', and 'Plugins and Extensions'. At the bottom of the sidebar are 'Advanced Options' and 'Apply Configuration'. The main area is titled 'Accounts and Roles'. It contains a section for 'Root Account Password' with instructions to enter a secure password. There are two password input fields, both showing four dots. Below them, it says 'Password strength: Weak'. The next section is 'MySQL User Accounts' with instructions to create user accounts and assign roles. Below this is a table with three columns: 'MySQL Username', 'Host', and 'User Role'. There is one row with a user icon, the username 'adminprova', the host '%', and the role 'DB Admin'. To the right of the table are three buttons: 'Add User' (highlighted with a blue border), 'Edit User', and 'Delete'. At the bottom right of the window are three buttons: '< Back', 'Next >', and 'Cancel'.

MySQL Installer
MySQL Server 5.7.21

Type and Networking
Type and Networking
Accounts and Roles
Windows Service
Plugins and Extensions
Advanced Options
Apply Configuration

Accounts and Roles

Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

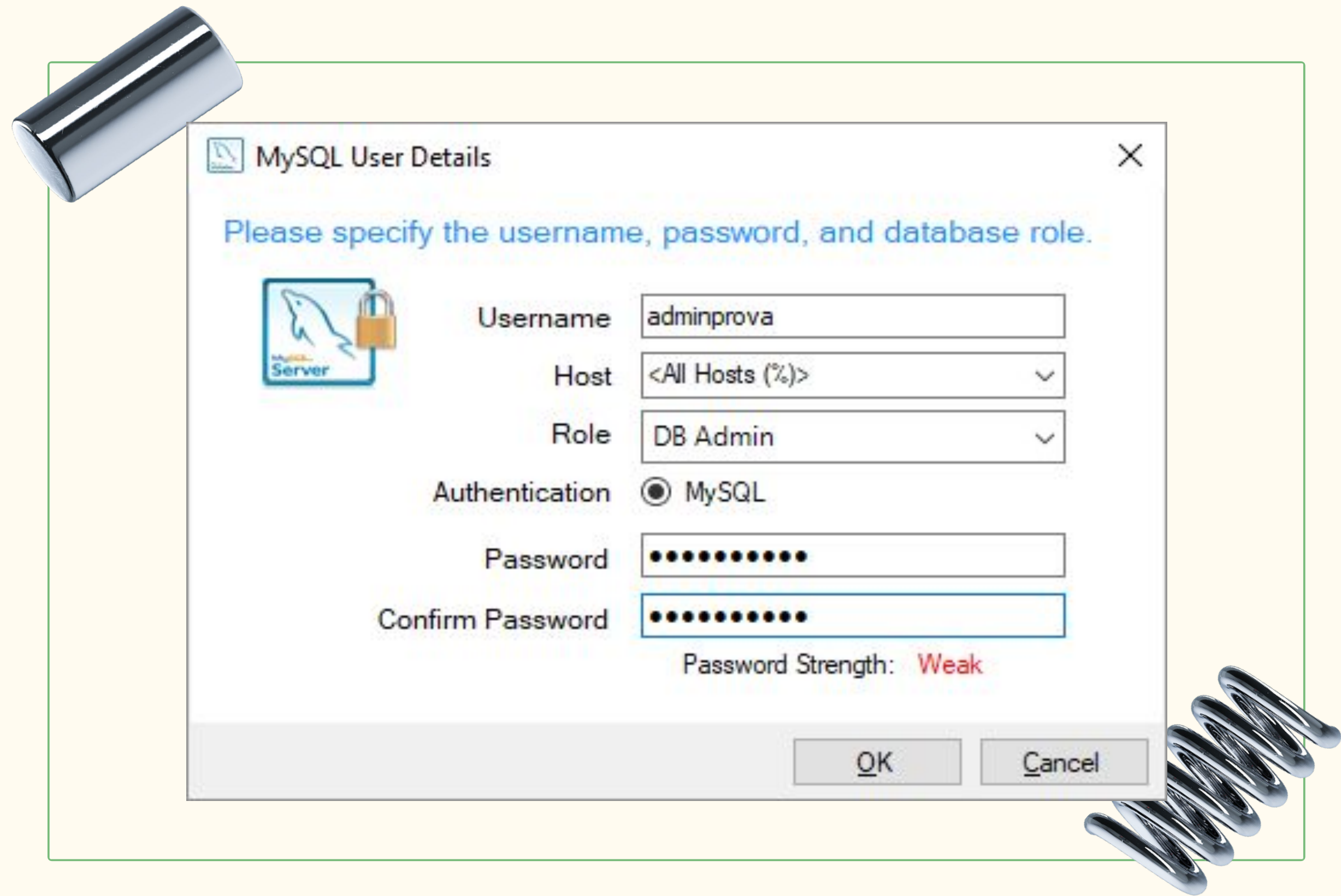
Password strength: **Weak**

MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL Username	Host	User Role
 adminprova	%	DB Admin

Add User
Edit User
Delete

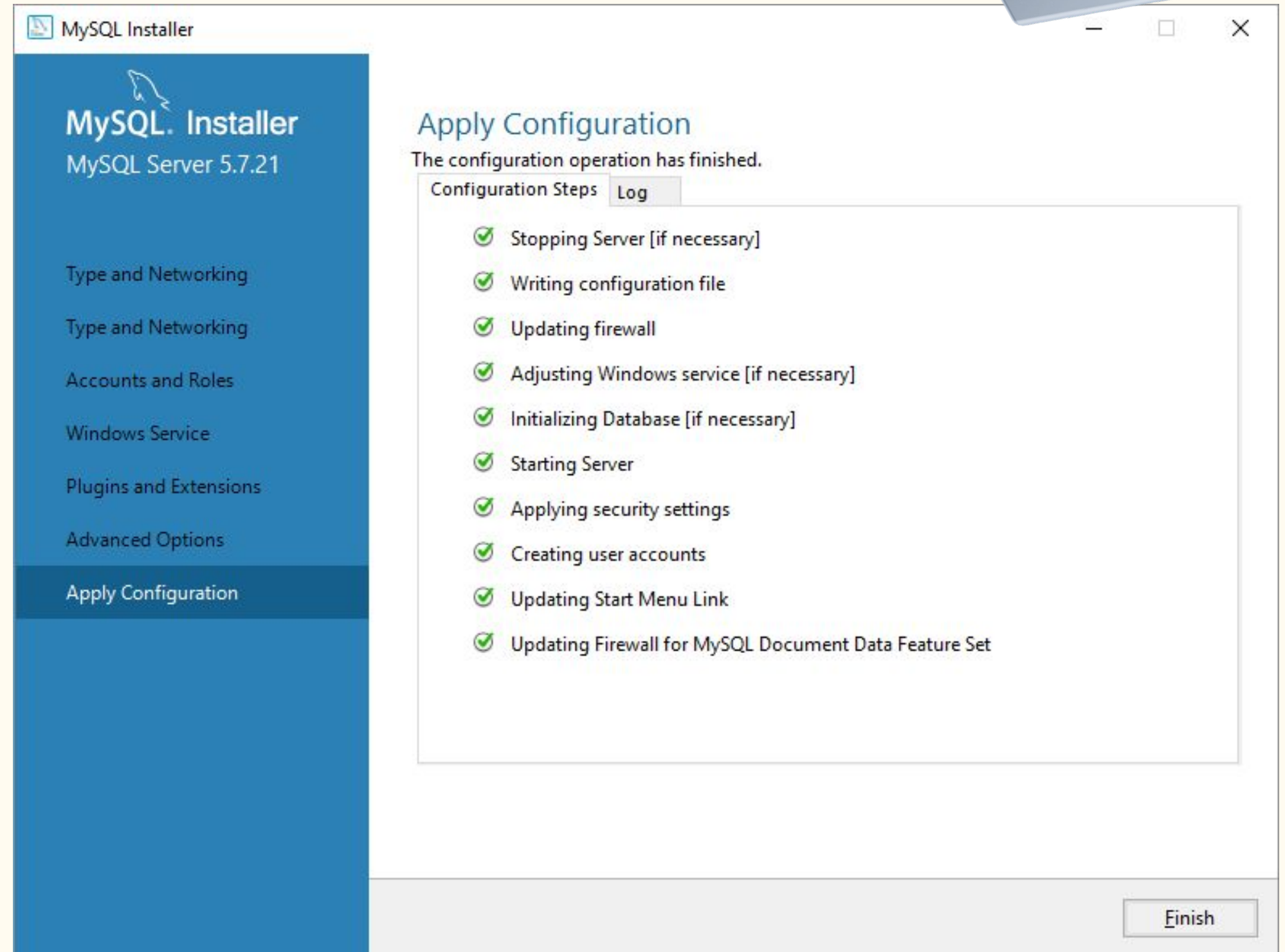
< Back Next > Cancel



Premi Next per accedere al passo successivo.

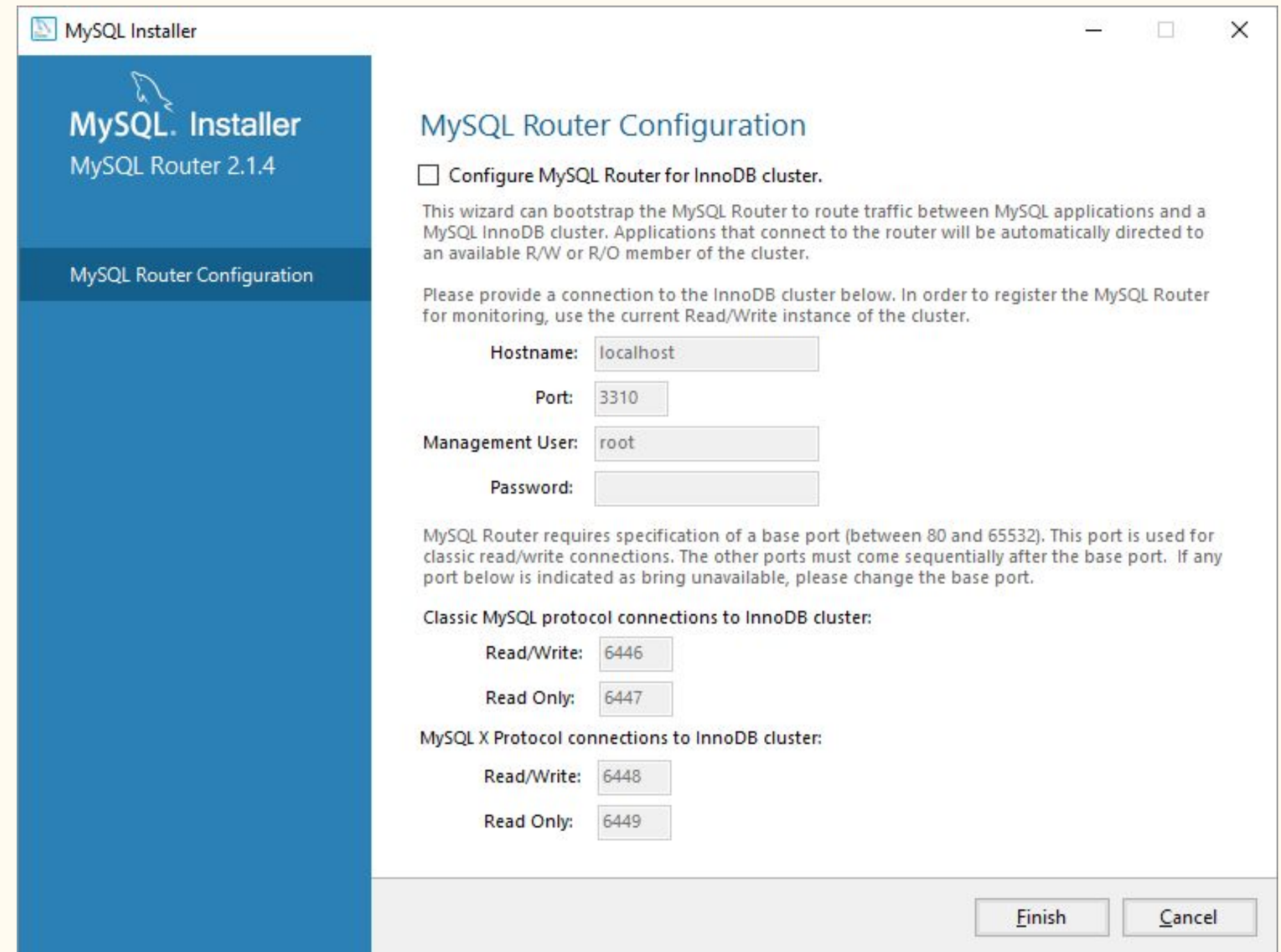
Apply Configuration:

- L'ultimo passaggio della configurazione, consiste nell'applicare tutto ciò che hai specificato fin qui.
- Assicurati che dopo aver premuto Execute, il risultato finale non presenti errori e sia simile all'immagine qui sopra.
- Se è tutto OK, direi che puoi passare al passo successivo, premendo Finish.



MySQL Router Configuration:

- In questo pannello ti verrà chiesto se vuoi attivare MySQL Router:
- Se non sai di che cosa sto parlando, non importa. Puoi semplicemente lasciare MySQL Router disabilitato e premere Finish.



The image shows a screenshot of the MySQL Installer application, specifically the 'MySQL Router Configuration' window. The window has a blue sidebar on the left with the MySQL logo and the text 'MySQL Installer' and 'MySQL Router 2.1.4'. The main area is white and contains the following text and fields:

MySQL Router Configuration

☐ Configure MySQL Router for InnoDB cluster.

This wizard can bootstrap the MySQL Router to route traffic between MySQL applications and a MySQL InnoDB cluster. Applications that connect to the router will be automatically directed to an available R/W or R/O member of the cluster.

Please provide a connection to the InnoDB cluster below. In order to register the MySQL Router for monitoring, use the current Read/Write instance of the cluster.

Hostname:

Port:

Management User:

Password:

MySQL Router requires specification of a base port (between 80 and 65532). This port is used for classic read/write connections. The other ports must come sequentially after the base port. If any port below is indicated as bring unavailable, please change the base port.

Classic MySQL protocol connections to InnoDB cluster:

Read/Write:

Read Only:

MySQL X Protocol connections to InnoDB cluster:

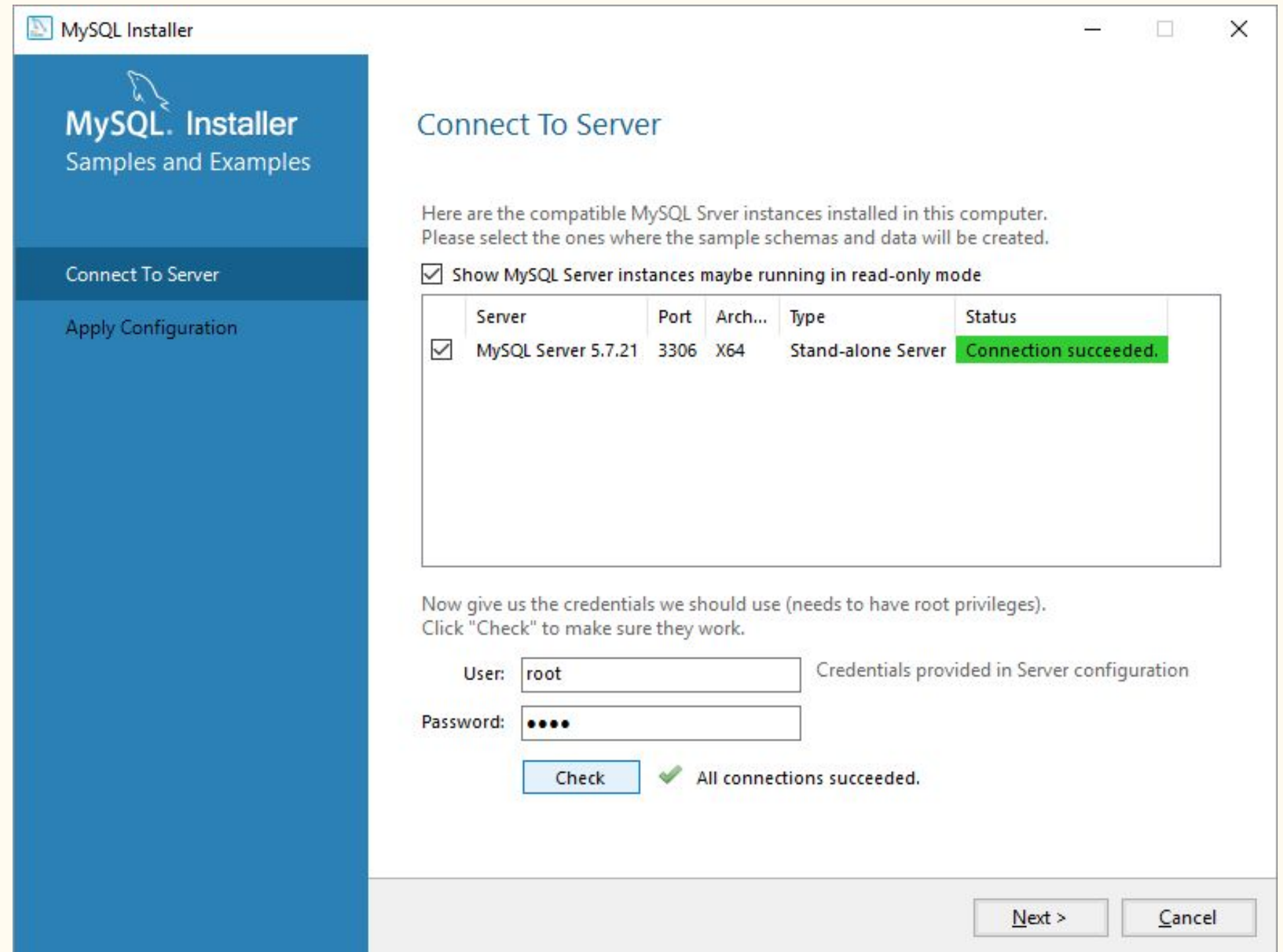
Read/Write:

Read Only:

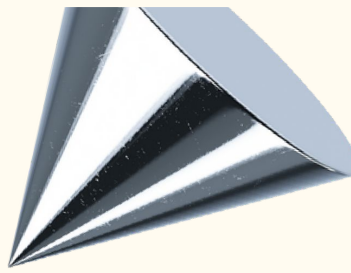
At the bottom right, there are two buttons: 'Finish' and 'Cancel'.

Connect to Server:

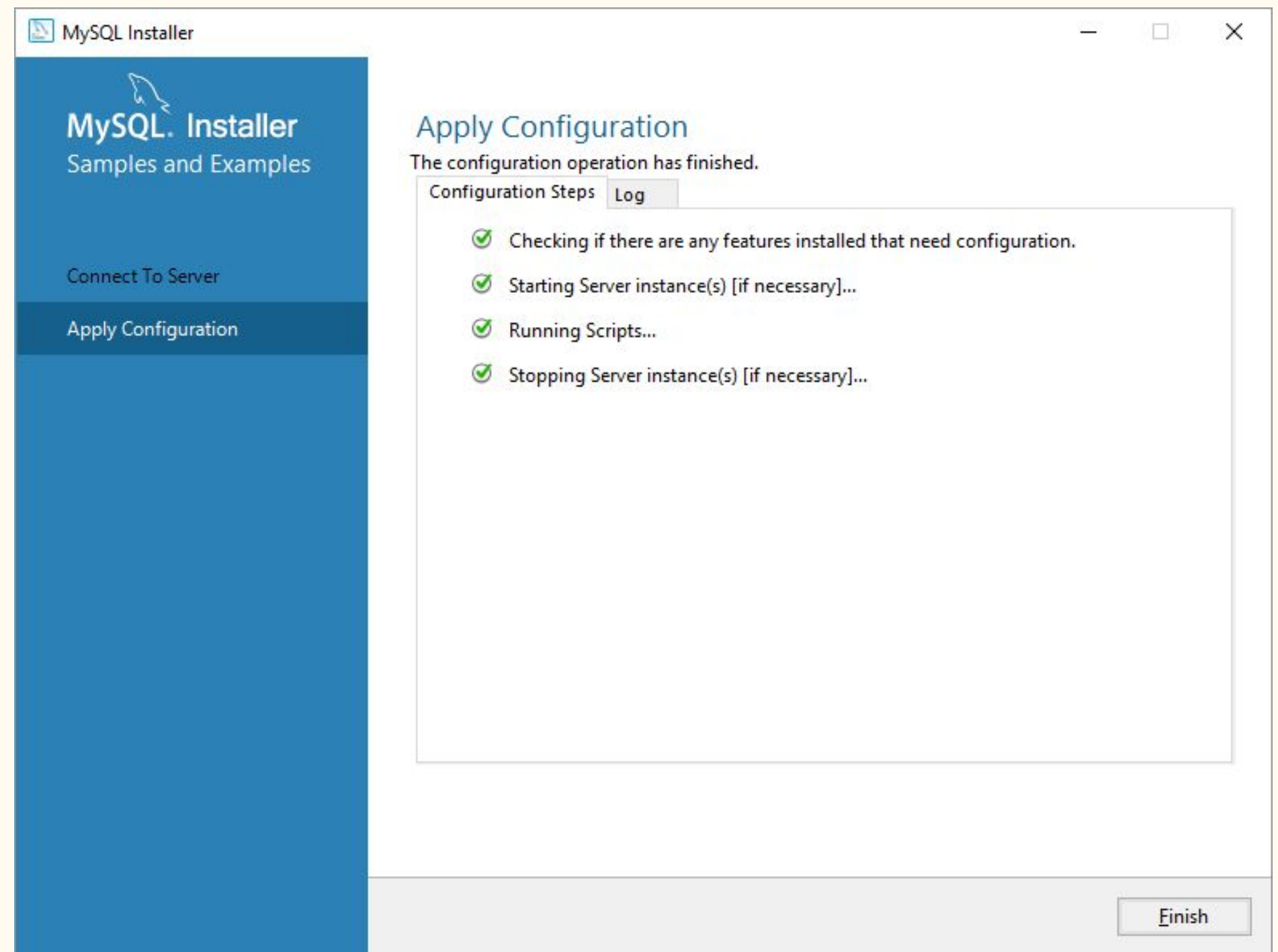
- Ora che tutto è installato, sarebbe utile verificare se la connessione al server funziona correttamente.
- Inserendo le credenziali degli account precedentemente definiti, puoi verificare la tua connessione al database.
- Se il test di connessione non ha problemi ti verrà segnalato con: Connection succeeded , ad indicare che tutto è andato bene.



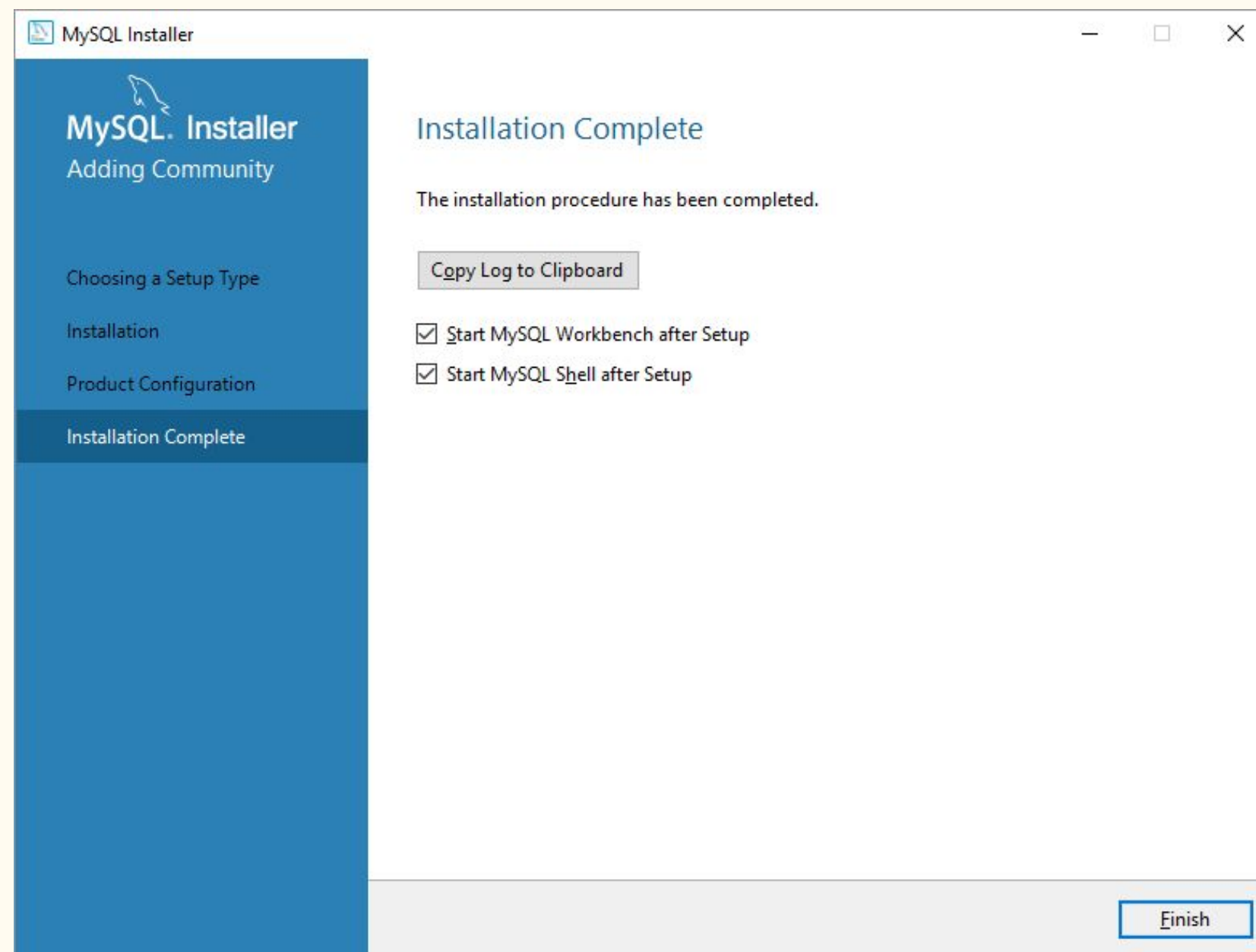
Apply Configuration (di nuovo):



- In modo che tutti i servizi vengano riavviati con le nuove definizioni.
- Premi il tasto Finish per passare all'ultima schermata.



Installation Complete:



Premi Finish per chiudere l'installatore ed avviare un terminale MySQL e Mysql Workbench, lo strumento di sviluppo per MySQL.

An abstract graphic featuring a thick, glossy, light grey liquid-like substance that flows from the top left, loops around, and then flows down towards the bottom right corner. The liquid has a highly reflective surface with bright highlights and shadows, giving it a three-dimensional, viscous appearance. The background is a solid, light cream color.

PROGETTO:

Login:

Questo codice HTML è un modulo di login. Quando un utente visita la pagina, vedrà due campi di input, uno per il nome utente e uno per la password. Entrambi i campi devono essere compilati per poter inviare il modulo, grazie all'attributo "required". Una volta che l'utente ha inserito le sue credenziali e ha cliccato sul pulsante di invio, i dati vengono inviati al server per essere elaborati da un file chiamato "login.py". Questo è fatto attraverso una richiesta HTTP POST, che invia i dati come corpo della richiesta, piuttosto che come parte dell'URL. Inoltre, il codice include un collegamento a un foglio di stile esterno ("style.css") per la formattazione e la presentazione della pagina.

```
login.html - Blocco note
File Modifica Formato Visualizza ?
<!DOCTYPE html>
<html lang="it">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="style.css">
  <title>Login</title>
</head>
<body>
  <form action="login.py" method="post">
    <h2>Login</h2>
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
    <input type="submit" value="Login">
  </form>
</body>
</html>
|
```


Questo script Python è progettato per gestire il processo di autenticazione di un'applicazione web. Inizialmente, stabilisce una connessione con un database MySQL e recupera le credenziali di accesso inviate dall'utente attraverso un modulo web. Successivamente, esegue due query SQL per cercare le credenziali dell'utente nelle tabelle 'utente' (per gli amministratori) e 'cliente' del database.

```
login.py - C:\inetpub\wwwroot\Agenzie\login.py (3.12.0)
File Edit Format Run Options Window Help
# La libreria cgi viene importata per gestire i dati del modulo inviati dall'ute
import cgi

# La libreria mysql.connector viene importata per connettersi al database MySQL.
import mysql.connector

print('Content-type: text/html\n\n')

# Connessione al database
cnx = mysql.connector.connect(user='root', password='1405FilippoSor!',
                              host='127.0.0.1',
                              database='agenzia')

# Questa linea stabilisce una connessione al database MySQL.
cursor = cnx.cursor()

# Form di inserimento credenziali
form = cgi.FieldStorage() # Questa linea crea un oggetto FieldStorage che contie

# Recupero dei dati dal form
username = form.getvalue('username')
password = form.getvalue('password') # Queste linee estraggono i valori di usern

# Query per cercare l'utente admin nel database
cerca_admin = "SELECT * FROM utente WHERE username = %s AND passw = %s"
admin = (username, password)

# Query per cercare l'utente cliente nel database
cerca_cliente = "SELECT * FROM cliente WHERE username = %s AND passw = %s"
cliente = (username, password)

# Esecuzione della query per admin
cursor.execute(cerca_admin, admin)
result_admin = cursor.fetchone()

# Queste linee eseguono le query SQL e o

# Esecuzione della query per cliente
cursor.execute(cerca_cliente, cliente)
result_cliente = cursor.fetchone()
```



```
login.py - C:\inetpub\wwwroot\Agenzie\login.py (3.12.0)
File Edit Format Run Options Window Help
if result_admin:
    # Se l'utente è un admin, viene visualizzata una pagina con vari link per le
    print('<html>')
    print('<head>')
    print('<title>Login Riuscito</title>')
    print('<link rel="stylesheet" type="text/css" href="style.css">')
    print('</head>')
    print('<body>')
    print('<h2>Login Riuscito! Accesso Consentito come Admin</h2>')
    print('<p>Clicca <a href="http://127.0.0.1:90/cliente.py">qui</a> per accede
    print('<p>Clicca <a href="http://127.0.0.1:90/admin.html">qui</a> per accede
    print('<p>Clicca <a href="http://127.0.0.1:90/Registrazione_Cliente.html">qu
    print('<div style="text-align:center;"><a href="http://127.0.0.1:90/login.ht
    print('</body>')
    print('</html>')
elif result_cliente:
    # Se l'utente è un cliente, viene reindirizzato alla pagina del cliente.
    print('<html>')
    print('<head>')
    print('<title>Login Riuscito</title>')
    print('<link rel="stylesheet" type="text/css" href="style.css">')
    print('<meta http-equiv="refresh" content="0;url=http://127.0.0.1:90/cliente
    print('</head>')
    print('<body>')
    print('<h2>Login Riuscito! Accesso Consentito come Cliente</h2>')
    print('<p>Clicca <a href="http://127.0.0.1:90/cliente.py">qui</a> per accede
    print('<div style="text-align:center;"><a href="http://127.0.0.1:90/login.ht
    print('</body>')
    print('</html>')
else:
    # Se le credenziali non corrispondono a nessun record nel database, informa
    print('<html>')
    print('<head>')
    print('<title>Login Fallito</title>')
    print('<link rel="stylesheet" type="text/css" href="style.css">')
    print('</head>')
    print('<body>')
    print('<h2>Login Fallito! Accesso Negato</h2>')
    print('<div style="text-align:center;"><a href="http://127.0.0.1:90/login.ht
    print('</body>')
    print('</html>')
```

Se le credenziali corrispondono a un record nella tabella 'utente', l'utente viene riconosciuto come amministratore e viene indirizzato a una pagina con vari link per le funzionalità dell'amministratore. Se invece le credenziali corrispondono a un record nella tabella 'cliente', l'utente viene riconosciuto come cliente e viene reindirizzato alla pagina del cliente.

Nel caso in cui le credenziali non corrispondano a nessun record nel database, l'utente viene informato che l'accesso è fallito. Infine, il cursore e la connessione al database vengono chiusi per liberare le risorse.

In sintesi, questo script gestisce l'autenticazione degli utenti, distinguendo tra amministratori e clienti, e reindirizza gli utenti alla pagina appropriata in base al loro ruolo. Se le credenziali fornite non sono valide, informa l'utente che l'accesso è fallito.

```
else:
    # Se le credenziali non corrispondono a nessun record nel database, informa
    print('<html>')
    print('<head>')
    print('<title>Login Fallito</title>')
    print('<link rel="stylesheet" type="text/css" href="style.css">')
    print('</head>')
    print('<body>')
    print('<h2>Login Fallito! Accesso Negato</h2>')
    print('<div style="text-align:center;"><a href="http://127.0.0.1:90/login.ht
    print('</body>')
    print('</html>')

# Chiusura della connessione
cursor.close()
cnx.close()      # Queste linee chiudono il cursore e la connessione al database
```


Client:

Questo script Python è progettato per gestire la ricerca di immobili in un'applicazione web. Inizialmente, stabilisce una connessione con un database MySQL e recupera i filtri di ricerca inviati dall'utente attraverso un modulo web. Successivamente, costruisce una query SQL in base a questi filtri e la esegue per recuperare i risultati.

```
Cliente.py - C:\inetpub\wwwroot\Agenzie\Cliente.py (3.12.0)
File Edit Format Run Options Window Help
# La libreria cgi viene importata per gestire i dati del modulo inviati dall'ute
import cgi

# La libreria mysql.connector viene importata per connettersi al database MySQL.
import mysql.connector

print('Content-type: text/html\n\n') # Questa linea stampa l'intestazione HTTP n

# Connessione al database
# Questa linea stabilisce una connessione al database MySQL.
cnx = mysql.connector.connect(user='root', password='1405FilippoSor!',
                              host='127.0.0.1',
                              database='agenzia')

cursor = cnx.cursor() # Questa linea crea un cursore che può essere utilizzato p

# Recupera i dati dal modulo di ricerca
form = cgi.FieldStorage() # Questa linea crea un oggetto FieldStorage che contie

ascensore_filter = form.getvalue('ascensore', 'Tutti')
localita_filter = form.getvalue('localita', 'Tutti') # Queste linee estraggono i

# Costruisci la query in base ai filtri
query_params = () # Questa linea inizializza una tupla vuota che conterrà i para

# Verifica il filtro sull'ascensore
if ascensore_filter in ['Si', 'No']:
    # Se il filtro ascensore è impostato su 'Si' o 'No', la query selezionerà so
    query_immobili = """
        SELECT
            i.id_imm,
            i.indirizzo,
            i.nvani,
            i.metratura,
            i.piano,
            i.ascensore,
            i.prezzo,
            i.Venduto,
            a.Ind AS agenzia_nome,
            l.nome AS localita_nome
        FROM
```

```

Cliente.py - C:\inetpub\wwwroot\Agenzie\Cliente.py (3.12.0)
File Edit Format Run Options Window Help

        FROM
            immobile i
        JOIN agenzia a ON i.agenzia = a.id_agenzia
        JOIN localita l ON i.localita = l.id_loc
        WHERE i.ascensore = %s
    """
    query_params = (ascensore_filter,)
else:
    # Se il filtro ascensore non è impostato su 'Si' o 'No', la query selezioner
    query_immobili = """
        SELECT
            i.id_imm,
            i.indirizzo,
            i.nvani,
            i.metratura,
            i.piano,
            i.ascensore,
            i.prezzo,
            i.Venduto,
            a.Ind AS agenzia_nome,
            l.nome AS localita_nome
        FROM
            immobile i
        JOIN agenzia a ON i.agenzia = a.id_agenzia
        JOIN localita l ON i.localita = l.id_loc
    """

# Aggiungi il filtro sulla località
if localita_filter != 'Tutti':
    # Se il filtro località è impostato su un valore diverso da 'Tutti', la quer
    query_immobili += " AND l.nome = %s"
    query_params += (localita_filter,)

# Esecuzione della query per immobili
cursor.execute(query_immobili, query_params) # Questa linea esegue la query SQL.

immobili_data = cursor.fetchall() # Questa linea recupera tutti i risultati dell

# Stampare la form HTML per la ricerca
print('<html>')
print('<head>')

```

Se ci sono risultati, li stampa in una tabella HTML. Ogni riga della tabella rappresenta un immobile e include dettagli come l'ID, l'indirizzo, il numero di vani, la metratura, il piano, la presenza dell'ascensore, il prezzo, lo stato di vendita, il nome dell'agenzia e la località.

Se non ci sono risultati, viene visualizzato un messaggio che informa l'utente che non sono stati trovati dati. Infine, il cursore e la connessione al database vengono chiusi per liberare le risorse.

Inoltre, il modulo di ricerca viene stampato nuovamente, permettendo all'utente di effettuare una nuova ricerca senza dover ricaricare la pagina. I filtri di ricerca includono la presenza dell'ascensore e la località, e l'utente può selezionare i valori desiderati da un menu a discesa.

```
Cliente.py - C:\inetpub\wwwroot\Agenzie\Cliente.py (3.12.0)
File Edit Format Run Options Window Help

# Stampare la form HTML per la ricerca
print('<html>')
print('<head>')
print('<title>Ricerca Immobili</title>')
print('<link rel="stylesheet" type="text/css" href="style.css">')
print('</head>')
print('<body>')
print('<h2>Ricerca Immobili</h2>')

print('<form method="post" action="">')
print('<label for="ascensore">Ascensore:</label>')
print('<select name="ascensore">')
print('<option value="Tutti" selected>Tutti</option>')
print('<option value="Si" {}>Si</option>'.format('selected' if ascensore_filter
print('<option value="No" {}>No</option>'.format('selected' if ascensore_filter
print('</select>')

print('<label for="localita">Località:</label>')
print('<select name="localita">')
print('<option value="Tutti" selected>Tutti</option>')
print('<option value="Torino" {}>Torino</option>'.format('selected' if localita_
print('<option value="Alessandria" {}>Alessandria</option>'.format('selected' if
print('<option value="Asti" {}>Asti</option>'.format('selected' if localita_filt
print('</select>')

print('<input type="submit" value="Ricerca">')
print('</form>')

print('<h2>Risultati della Query</h2>') # Queste linee stampano un modulo HTML c

# Verifica se ci sono risultati nella ricerca
if immobili_data:
    # Se ci sono risultati, vengono stampati in una tabella HTML.
    print('<table border="1">')
    print('<tr>')
    print('<th>ID</th>')
    print('<th>Indirizzo</th>')
    print('<th>Numero Vani</th>')
    print('<th>Mettratura</th>')
    print('<th>Piano</th>')
    print('<th>Ascensore</th>')
    print('</tr>')
    print('<tbody>')
    for row in immobili_data:
        print('<tr>')
        print('<td>{}</td>'.format(row['ID']))
        print('<td>{}</td>'.format(row['Indirizzo']))
        print('<td>{}</td>'.format(row['Numero Vani']))
        print('<td>{}</td>'.format(row['Mettratura']))
        print('<td>{}</td>'.format(row['Piano']))
        print('<td>{}</td>'.format(row['Ascensore']))
        print('</tr>')
    print('</tbody>')
    print('</table>')
```



```
Cliente.py - C:\inetpub\wwwroot\Agenzie\Cliente.py (3.12.0)
File Edit Format Run Options Window Help
# Se ci sono risultati, vengono stampati in una tabella HTML.
print('<table border="1">')
print('<tr>')
print('<th>ID</th>')
print('<th>Indirizzo</th>')
print('<th>Numero Vani</th>')
print('<th>Metratura</th>')
print('<th>Piano</th>')
print('<th>Ascensore</th>')
print('<th>Prezzo</th>')
print('<th>Venduto</th>')
print('<th>Agenzia</th>')
print('<th>Località</th>')
print('</tr>')

for row in immobili_data:
    # Ogni riga di dati viene stampata come una riga della tabella.
    print('<tr>')
    print('<td>{}</td>'.format(row[0]))
    print('<td>{}</td>'.format(row[1]))
    print('<td>{}</td>'.format(row[2]))
    print('<td>{}</td>'.format(row[3]))
    print('<td>{}</td>'.format(row[4]))
    print('<td>{}</td>'.format(row[5]))
    print('<td>{}</td>'.format(row[6]))
    print('<td>{}</td>'.format(row[7]))
    print('<td>{}</td>'.format(row[8]))
    print('<td>{}</td>'.format(row[9]))
    print('</tr>')

print('</table>')
else:
    # Se non ci sono risultati, viene stampato un messaggio che informa l'utente
    print('<p>Nessun dato trovato.</p>')

print('</body>')
print('</html>')

# Chiusura della connessione
cursor.close()
cnx.close() # Queste linee chiudono il cursore e la connessione al database.
```

In sintesi, questo script gestisce la ricerca di immobili in un'applicazione web, permettendo all'utente di filtrare i risultati in base alla presenza dell'ascensore e alla località. I risultati vengono visualizzati in una tabella HTML per una facile lettura, e se non ci sono risultati, viene visualizzato un messaggio appropriato. Infine, il modulo di ricerca viene stampato nuovamente, permettendo all'utente di effettuare una nuova ricerca immediatamente.

Administrator:

Questo codice HTML rappresenta un modulo di inserimento per un amministratore di un'applicazione web di immobili. L'amministratore può inserire dettagli su un immobile, come l'ID, l'indirizzo, il numero di vani, la metratura, il piano, la presenza dell'ascensore, il prezzo, lo stato di vendita, l'agenzia e la località. Una volta compilato il modulo, i dati vengono inviati a un file chiamato "admin.py" sul server per l'elaborazione.

In sintesi, questo modulo consente agli amministratori di inserire nuovi immobili nel database dell'applicazione web.

```
admin.html - Blocco note
File Modifica Formato Visualizza ?
<!DOCTYPE html>
<html lang="it">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="style.css">
  <title>Inserimento Immobili</title>
</head>
<body>
  <h2>Inserimento informazioni Amministratore</h2>
  <form action="admin.py" method="post">

    <label for="id">ID:</label>
    <input type="text" id="id" name="id" required><br>

    <label for="indirizzo">Indirizzo:</label>
    <input type="text" id="indirizzo" name="indirizzo" required><br>

    <label for="nvani">Numero Vani:</label>
    <input type="text" id="nvani" name="nvani" required><br>

    <label for="metratura">Metratura:</label>
    <input type="text" id="metratura" name="metratura" required><br>

    <label for="piano">Piano:</label>
    <input type="text" id="piano" name="piano" required><br>

    <label for="ascensore">Ascensore:</label>
    <select id="ascensore" name="ascensore" required>
      <option value="Si">Si</option>
      <option value="No">No</option>
    </select><br>

  </form>
</body>
</html>
```

Windows (CRLF) Linea 55, colonna 17 100%


```
admin.py - C:\inetpub\wwwroot\Agenzie\admin.py (3.12.0)
File Edit Format Run Options Window Help
# La libreria cgi viene importata per gestire i dati del modulo inviati dall'utente.
import cgi

# La libreria mysql.connector viene importata per connettersi al database MySQL.
import mysql.connector

# Questa linea stampa l'intestazione HTTP necessaria per un documento HTML.
print('Content-type: text/html\n\n')

# Connessione al database
# Questa linea stabilisce una connessione al database MySQL.
cnx = mysql.connector.connect(user='root', password='1405FilippoSor!',
                              host='127.0.0.1',
                              database='agenzia')

cursor = cnx.cursor() # Questa linea crea un cursore che può essere utilizzato per es

# Form di inserimento dati immobile
form = cgi.FieldStorage() # Questa linea crea un oggetto FieldStorage che contiene i

# Recupero dei dati dal form
id_immobile = form.getvalue('id')
indirizzo = form.getvalue('indirizzo')
nvani = form.getvalue('nvani')
metratura = form.getvalue('metratura')
piano = form.getvalue('piano')
ascensore = form.getvalue('ascensore')
prezzo = form.getvalue('prezzo')
venduto = form.getvalue('venduto')
agenzia_nome = form.getvalue('agenzia') # Nome dell'agenzia inserito dall'utente
localita_nome = form.getvalue('localita') # Nome della località inserito dall'utente

# Ottieni l'ID dell'agenzia dalla tabella "agenzia" usando il nome
cerca_agenzia = "SELECT id_agenzia FROM agenzia WHERE Ind = %s"
cursor.execute(cerca_agenzia, (agenzia_nome,))
result_agenzia = cursor.fetchone()
if result_agenzia:
    id_agenzia = result_agenzia[0]
else:
    # Puoi gestire un'agenzia non trovata in base alle tue esigenze
```

Questo script Python gestisce l'inserimento di nuovi immobili in un'applicazione web. Recupera i dati di un immobile inviati dall'amministratore attraverso un modulo web, cerca gli ID corrispondenti per l'agenzia e la località nel database, e poi inserisce il nuovo immobile nel database.

Se l'inserimento è riuscito, viene visualizzato un messaggio di successo. Se si verifica un errore durante l'inserimento, viene visualizzato un messaggio di errore. Infine, il cursore e la connessione al database vengono chiusi per liberare le risorse.

```
admin.py - C:\inetpub\wwwroot\Agenzie\admin.py (3.12.0)
File Edit Format Run Options Window Help

# Ottieni l'ID della località dalla tabella "localita" usando il nome
cerca_localita = "SELECT id_loc FROM localita WHERE nome = %s"
cursor.execute(cerca_localita, (localita_nome,))
result_localita = cursor.fetchone()

# Leggi completamente i risultati di cerca_localita prima di procedere
cursor.fetchall()

if result_localita:
    id_localita = result_localita[0]
else:
    # Puoi gestire una località non trovata in base alle tue esigenze
    id_localita = None

# Esecuzione della query solo se il form è stato inviato e l'azione è "Inserisci"
if form.getvalue('action_inserisci'):
    try:
        # Query di inserimento nella tabella "immobile"
        inserisci_immobile_query = """
            INSERT INTO immobile
            (id_imm, indirizzo, nvani, metratura, piano, ascensore, prezzo, Venduto,
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
        """
        dati_immobile = (id_immobile, indirizzo, nvani, metratura, piano, ascensore,
        cursor.execute(inserisci_immobile_query, dati_immobile)
        cnx.commit()

        # Dopo l'inserimento, puoi anche effettuare un redirect o mostrare un messagg
        print('<p>Dati inseriti con successo!</p>')
        print('<div style="text-align:center;"><a href="http://127.0.0.1:90/admin.htm
    except mysql.connector.Error as err:
        # Gestisci eventuali errori durante l'inserimento
        print('<p>Errore durante l'inserimento dei dati: {}</p>'.format(err))
        print('<div style="text-align:center;"><a href="http://127.0.0.1:90/admin.htm

# Chiusura della connessione
cursor.close()
cnx.close() # Queste linee chiudono il cursore e la connessione al database.
```

Registrazione Cliente:

Questo codice HTML rappresenta un modulo di registrazione per un utente in un'applicazione web. L'utente può inserire il proprio nome, username e password. Una volta compilato il modulo, i dati vengono inviati a un file chiamato "Registrazione_Cliente.py" sul server per l'elaborazione. In sintesi, questo modulo consente agli utenti di registrarsi nell'applicazione web.

```
Registrazione_Cliente.html - Blocco note
File Modifica Formato Visualizza ?
<!DOCTYPE html>
<html lang="it">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="style.css">
  <title>Inserimento Utente</title>
</head>
<body>
  <h2>Inserimento informazioni Utente</h2>
  <form action="Registrazione_Cliente.py" method="post">

    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome" required><br>

    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br>

    <label for="passw">Password:</label>
    <input type="password" id="passw" name="passw" required><br>

    <input type="submit" value="Inserisci" name="action_inserisci">

  </form>
</body>
</html>
```


Questo script Python è progettato per gestire la registrazione di nuovi utenti in un'applicazione web di immobili. Inizialmente, stabilisce una connessione con un database MySQL. Successivamente, recupera i dati dell'utente inviati attraverso un modulo web.

Questi dati includono il nome, l'username e la password dell'utente.

Il script genera poi un ID unico per il nuovo utente. Questo viene fatto recuperando l'ID più grande attualmente presente nella tabella dei clienti del database e incrementandolo di uno.

Una volta generato l'ID, il nuovo utente viene inserito nel database. Questo viene fatto eseguendo una query SQL di inserimento che aggiunge un nuovo record alla tabella dei clienti con i dati dell'utente e l'ID generato.

```
Registrazione_Cliente.py - C:\inetpub\wwwroot\Agenzie\Registrazione_Cliente.py (3.12.0)
File Edit Format Run Options Window Help

# La libreria cgi viene importata per gestire i dati del modulo inviati dall'ute
import cgi

# La libreria mysql.connector viene importata per connettersi al database MySQL
import mysql.connector

# Questa linea stampa l'intestazione HTTP necessaria per un documento HTML.
print('Content-type: text/html\n\n')

# Connessione al database
# Questa linea stabilisce una connessione al database MySQL.
cnx = mysql.connector.connect(user='root', password='1405FilippoSor!',
                              host='127.0.0.1',
                              database='agenzia')

cursor = cnx.cursor() # Questa linea crea un cursore che può essere utilizzato p

# Form di inserimento dati utente
form = cgi.FieldStorage() # Questa linea crea un oggetto FieldStorage che contie

# Recupero dei dati dal form
nome = form.getvalue('nome')
username = form.getvalue('username')
password = form.getvalue('passw') # Queste linee estraggono i valori di nome, us

# Esecuzione della query solo se il form è stato inviato e l'azione è "Inserisci"
if form.getvalue('action_inserisci'):
    try:
        # Ottieni l'ID più grande attualmente nella tabella
        cursor.execute("SELECT MAX(id) FROM cliente")
        max_id = cursor.fetchone()[0]

        # Se la tabella è vuota, imposta max_id a 0
        if max_id is None:
            max_id = 0

        # Incrementa l'ID per il nuovo record
        id = max_id + 1

        # Query di inserimento nella tabella "cliente"
```



```
Registrazione_Cliente.py - C:\inetpub\wwwroot\Agenzie\Registrazione_Cliente.py (3.12.0)
File Edit Format Run Options Window Help
username = form.getvalue('username')
password = form.getvalue('passw') # Queste linee estraggono i valori di nome, us

# Esecuzione della query solo se il form è stato inviato e l'azione è "Inserisci"
if form.getvalue('action_inserisci'):
    try:
        # Ottieni l'ID più grande attualmente nella tabella
        cursor.execute("SELECT MAX(id) FROM cliente")
        max_id = cursor.fetchone()[0]

        # Se la tabella è vuota, imposta max_id a 0
        if max_id is None:
            max_id = 0

        # Incrementa l'ID per il nuovo record
        id = max_id + 1

        # Query di inserimento nella tabella "cliente"
        inserisci_utente_query = """
            INSERT INTO `cliente`
            (`id`, `nome`, `username`, `passw`)
            VALUES (%s, %s, %s, %s)
        """
        dati_utente = (id, nome, username, password)

        cursor.execute(inserisci_utente_query, dati_utente)
        cnx.commit()

        # Dopo l'inserimento, puoi anche effettuare un redirect o mostrare un me
        print('<p>Dati inseriti con successo!</p>')
        print('<link rel="stylesheet" type="text/css" href="style.css">')
    except mysql.connector.Error as err:
        # Gestisci eventuali errori durante l'inserimento
        print('<p>Errore durante l'inserimento dei dati: {}</p>'.format(err))
        print('<link rel="stylesheet" type="text/css" href="style.css">')

# Chiusura della connessione
cursor.close()
cnx.close() # Queste linee chiudono il cursore e la connessione al database.
```

Se l'inserimento dei dati nel database è riuscito, il script stampa un messaggio di successo. Se si verifica un errore durante l'inserimento dei dati, il script stampa un messaggio di errore.

Infine, il cursore e la connessione al database vengono chiusi. Questo è importante per liberare le risorse del sistema e prevenire possibili problemi di sicurezza.

In sintesi, questo script gestisce l'intero processo di registrazione di un nuovo utente, dal recupero dei dati dell'utente all'inserimento di questi dati nel database. Gestisce anche eventuali errori che possono verificarsi durante questo processo.



Grazie per l'attenzione
