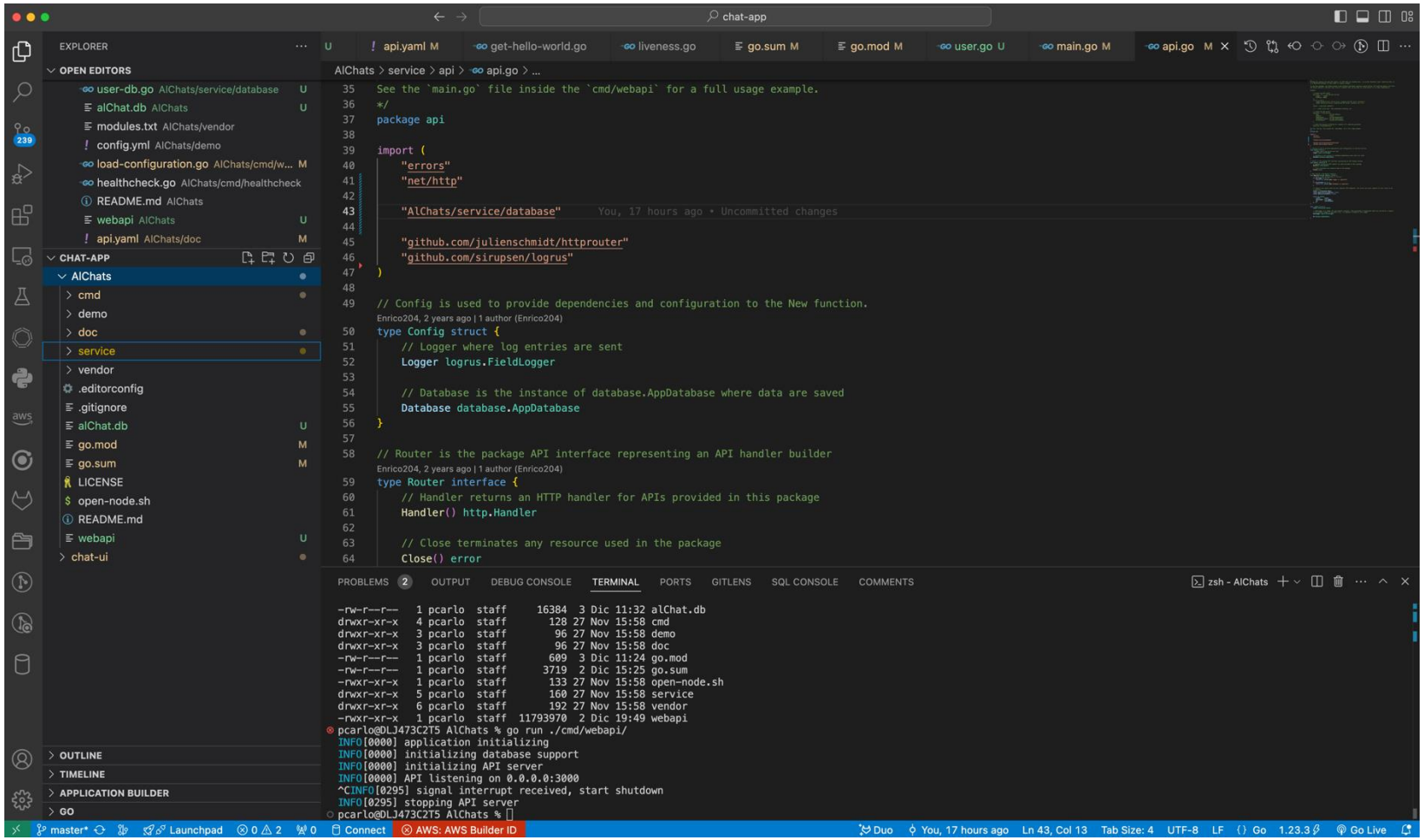
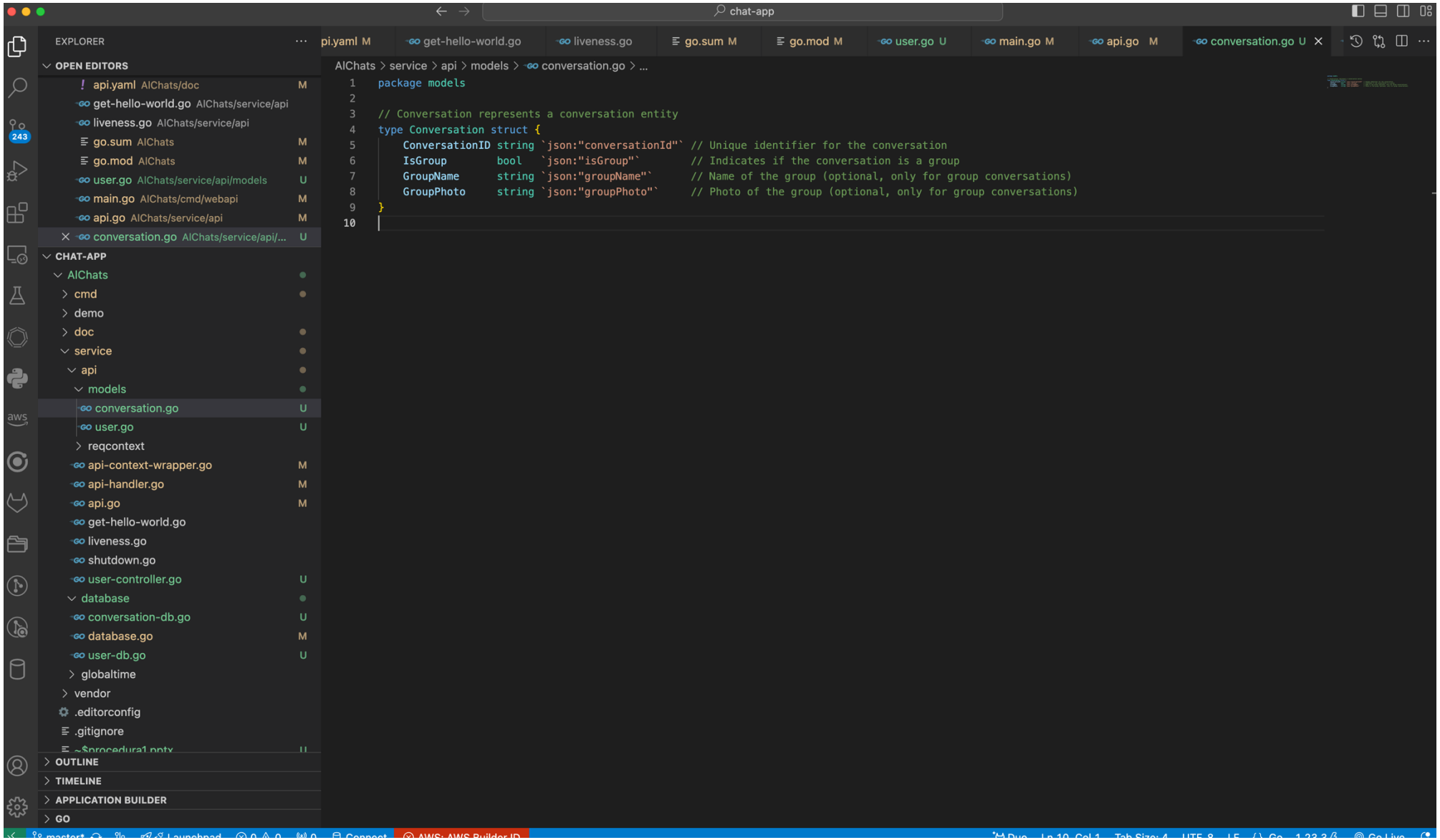
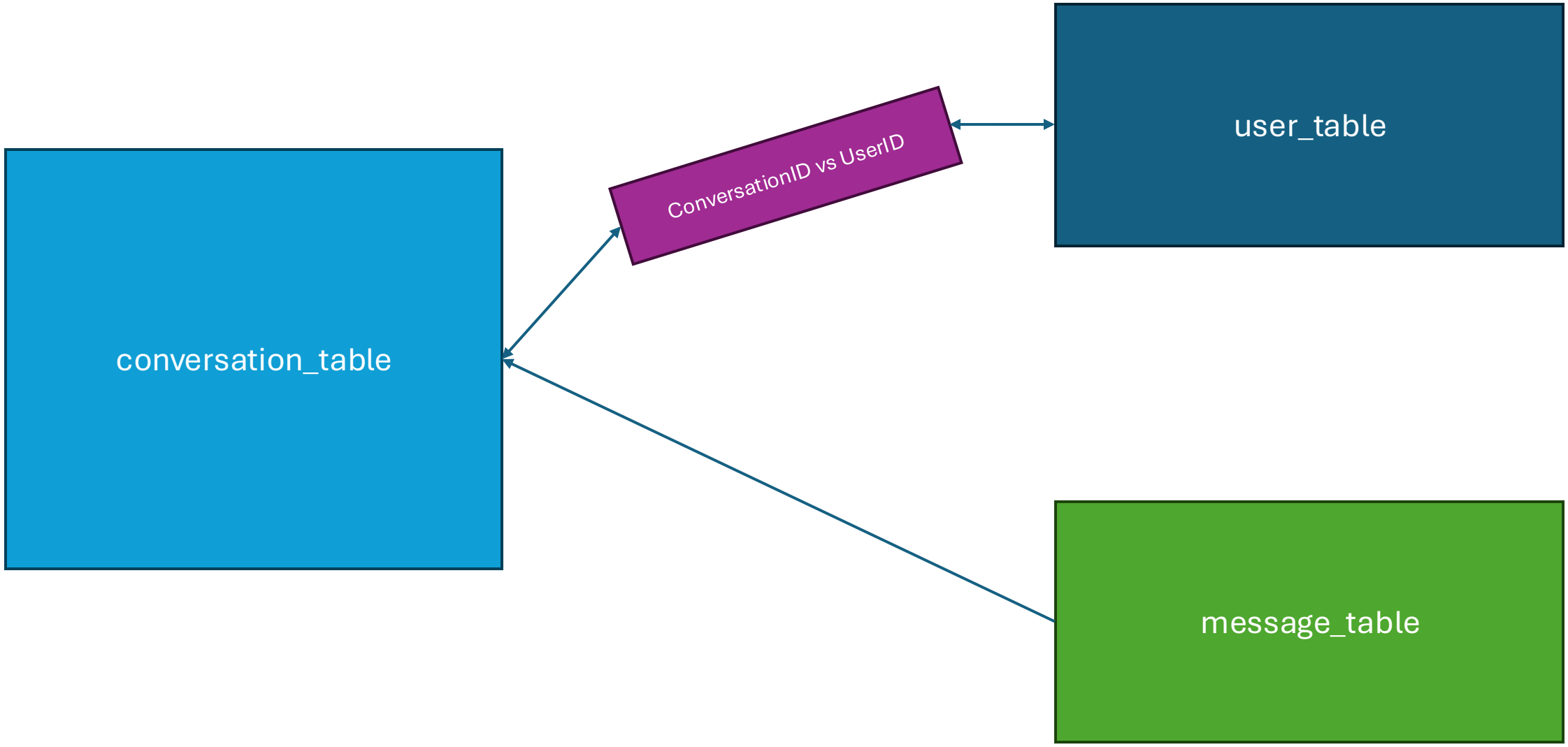
Come aggiungere un nuovo modello, le sue query Db e i sui endpoint http



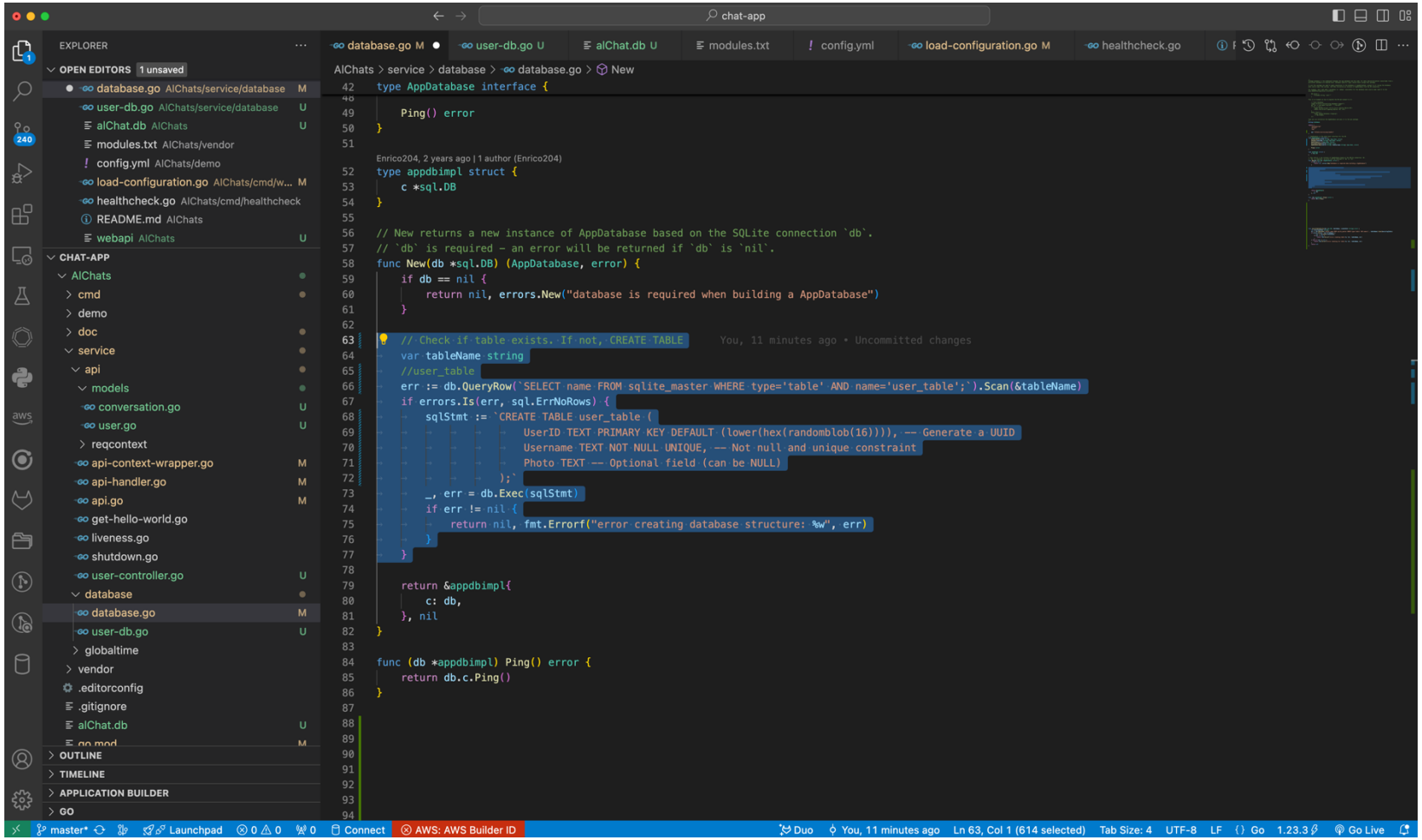
Create a Model Conversation



Design Database Tables for Conversation



DDL and Initialization for Conversation

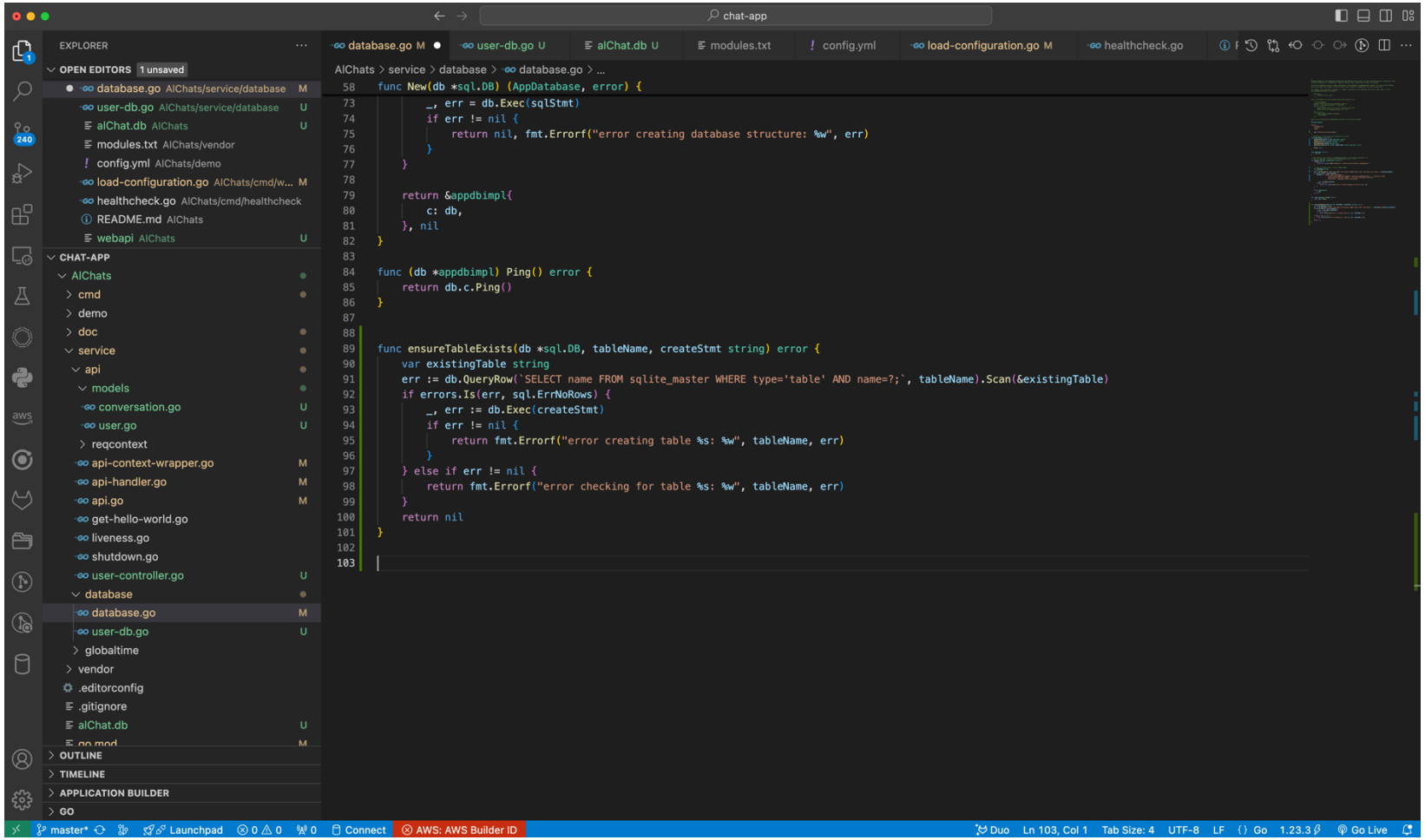


in questa sezione va aggiunta la verifica per la nuova tabella

Ma conviene usilizzare delle fuzioni ausiliarie per gestire molte tabelle

vediamo come nelle prossime slides

DDL and Initialization for Conversation



aggiungiamo la funzione ensureTableExists

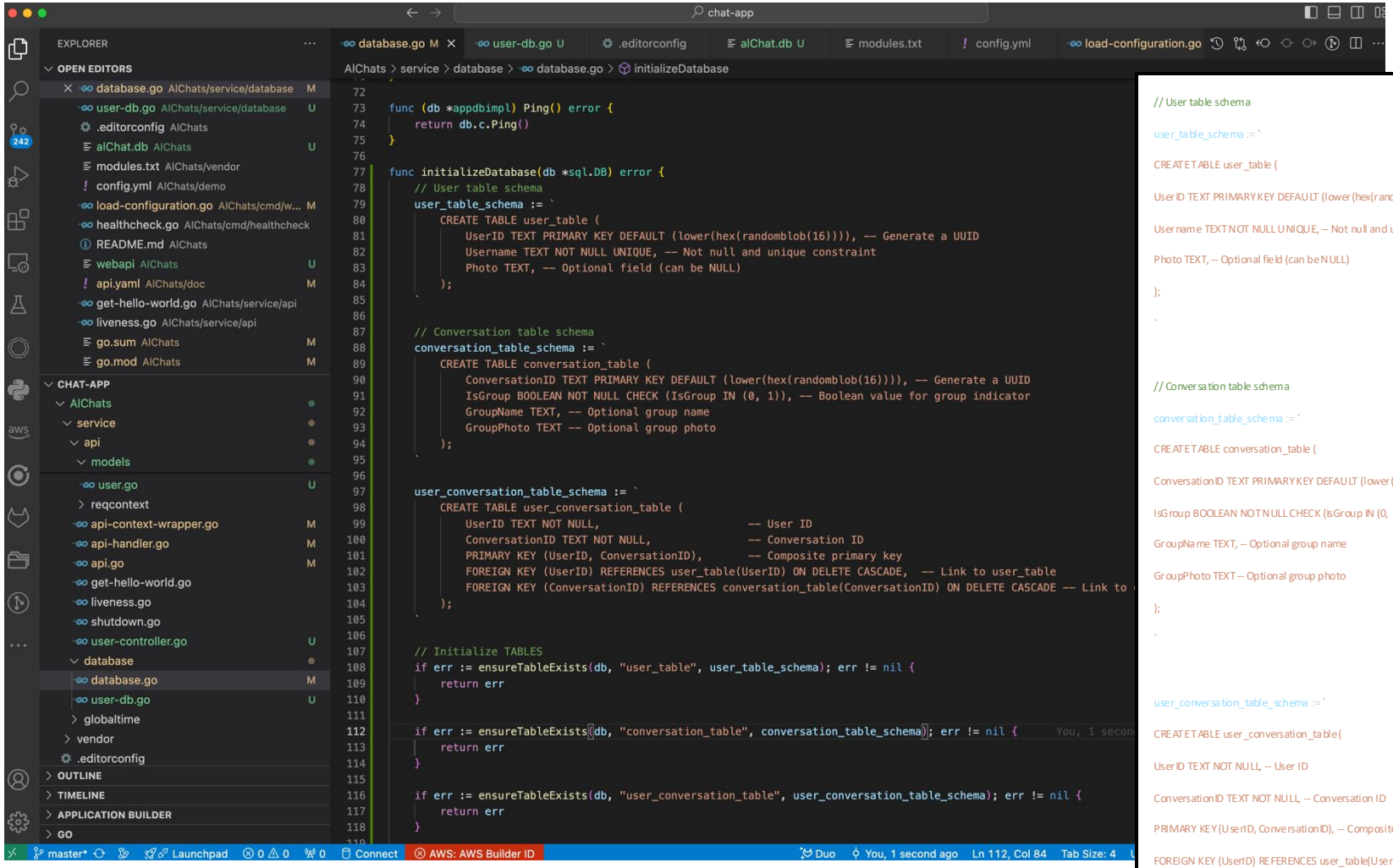
che verifica solo se una tabella é già eistente

DDL and Initialization for Conversation

aggiungiamo la funzione initializeDatabase

che crea le tabelle se non esistono

aggiungo una tabella per testire la relazione molti a molti

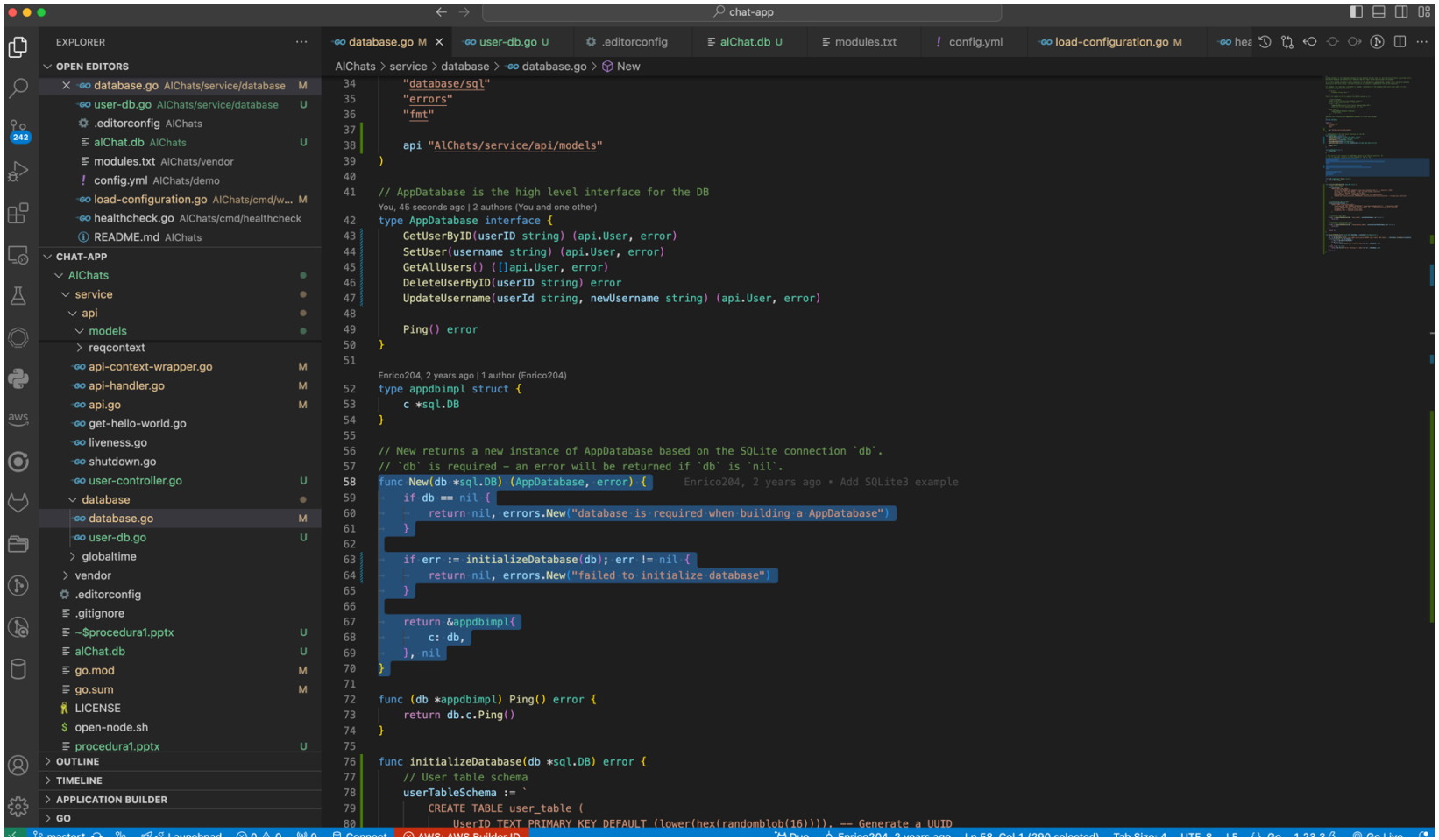


// User table schema
user_table_schema := `
CREATE TABLE user_table (
UserID TEXT PRIMARY KEY DEFAULT (lower(hex(randomblob(16)))),
Username TEXT NOT NULL UNIQUE, -- Not null and unique constra
Photo TEXT, -- Optional field (can be NULL)
);
`
// Conversation table schema
conversation_table_schema := `
CREATE TABLE conversation_table (
ConversationID TEXT PRIMARY KEY DEFAULT (lower(hex(randomblob(16)))), -- Generate a UUID
IsGroup BOOLEAN NOT NULL CHECK (IsGroup IN (0, 1)), -- Boolean value for group indicator
GroupName TEXT, -- Optional group name
GroupPhoto TEXT -- Optional group photo
);
`
user_conversation_table_schema := `
CREATE TABLE user_conversation_table (
UserID TEXT NOT NULL, -- User ID
ConversationID TEXT NOT NULL, -- Conversation ID
PRIMARY KEY (UserID, ConversationID), -- Composite primary key
FOREIGN KEY (UserID) REFERENCES user_table(UserID) ON DELETE CASCADE, -- Link to user_table
FOREIGN KEY (ConversationID) REFERENCES conversation_table(ConversationID) ON DELETE CASCADE -- Link to conversation_table
);
`

DDL and Initialization for Conversation

modifichiamo la funzione New
in modo da chiamare
le nostre funzioni

testiamo la nuova struttura



DDL and Initialization for Conversation

testiamo la nuova struttura

cancellazione db

```
pcarlo@DLJ473C2T5 chat-app % cd AlChats
pcarlo@DLJ473C2T5 AlChats % ls -al al*
-rw-r--r--  1 pcarlo  staff  16384  3 Dic 12:58 alChat.db
pcarlo@DLJ473C2T5 AlChats % rm alChat.db
pcarlo@DLJ473C2T5 AlChats % ls -al al*
zsh: no matches found: al*
pcarlo@DLJ473C2T5 AlChats %
```

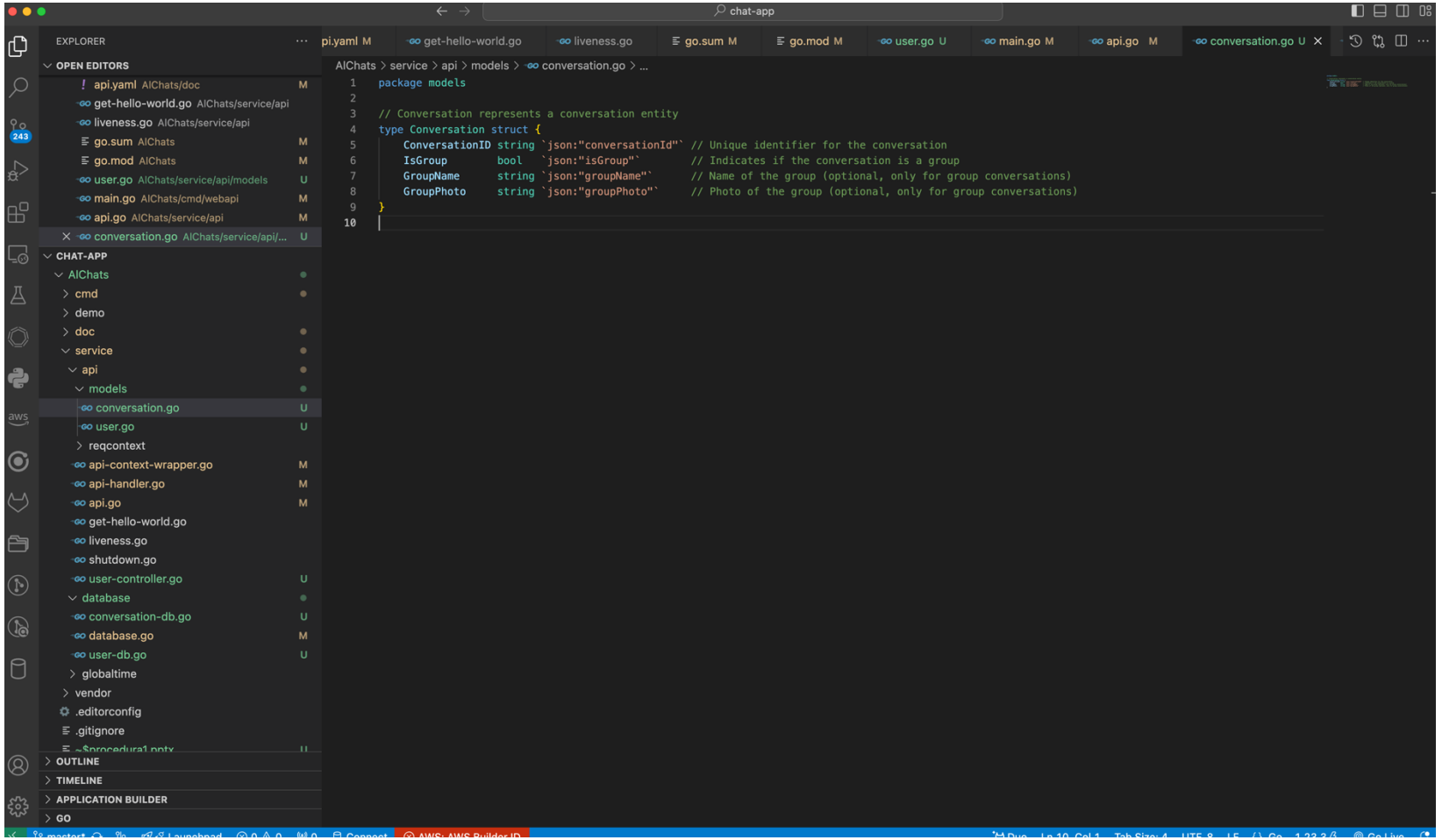
avvio programma

```
pcarlo@DLJ473C2T5 AlChats % go run ./cmd/webapi/
INFO[0000] application initializing
INFO[0000] initializing database support
INFO[0000] initializing API server
INFO[0000] API listening on 0.0.0.0:3000
```

check database

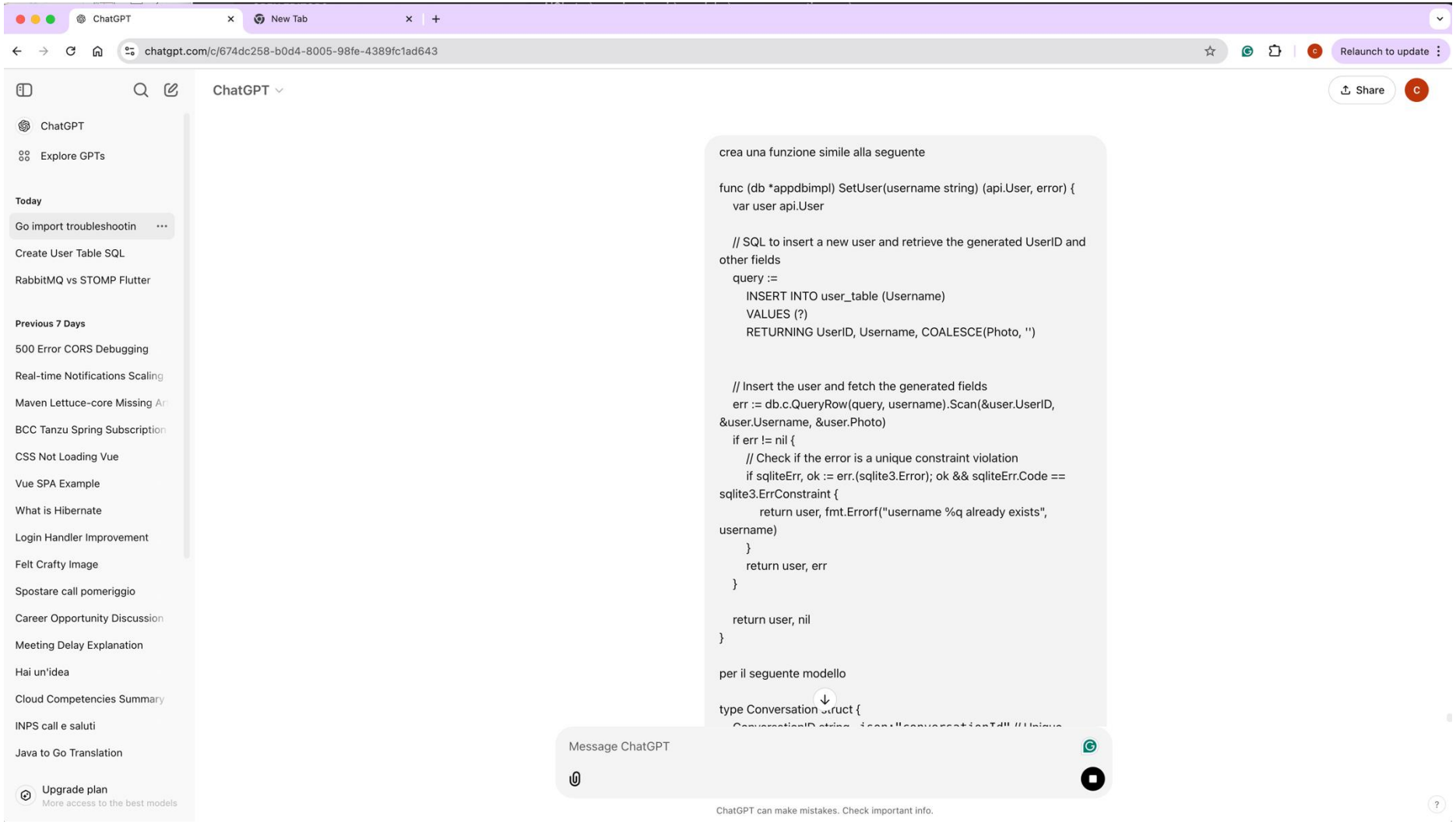
```
pcarlo@DLJ473C2T5 AlChats % ls -al al*
-rw-r--r--  1 pcarlo  staff  24576  3 Dic 13:40 alChat.db
pcarlo@DLJ473C2T5 AlChats % sqlite3 alChat.db
SQLite version 3.43.2 2023-10-10 13:08:14
Enter ".help" for usage hints.
sqlite> SELECT name FROM sqlite_master WHERE type='table';
user_table
conversation_table
user_conversation_table
sqlite>
```

Creazione funzioni DB per Conversation



creazione
modello per conversation

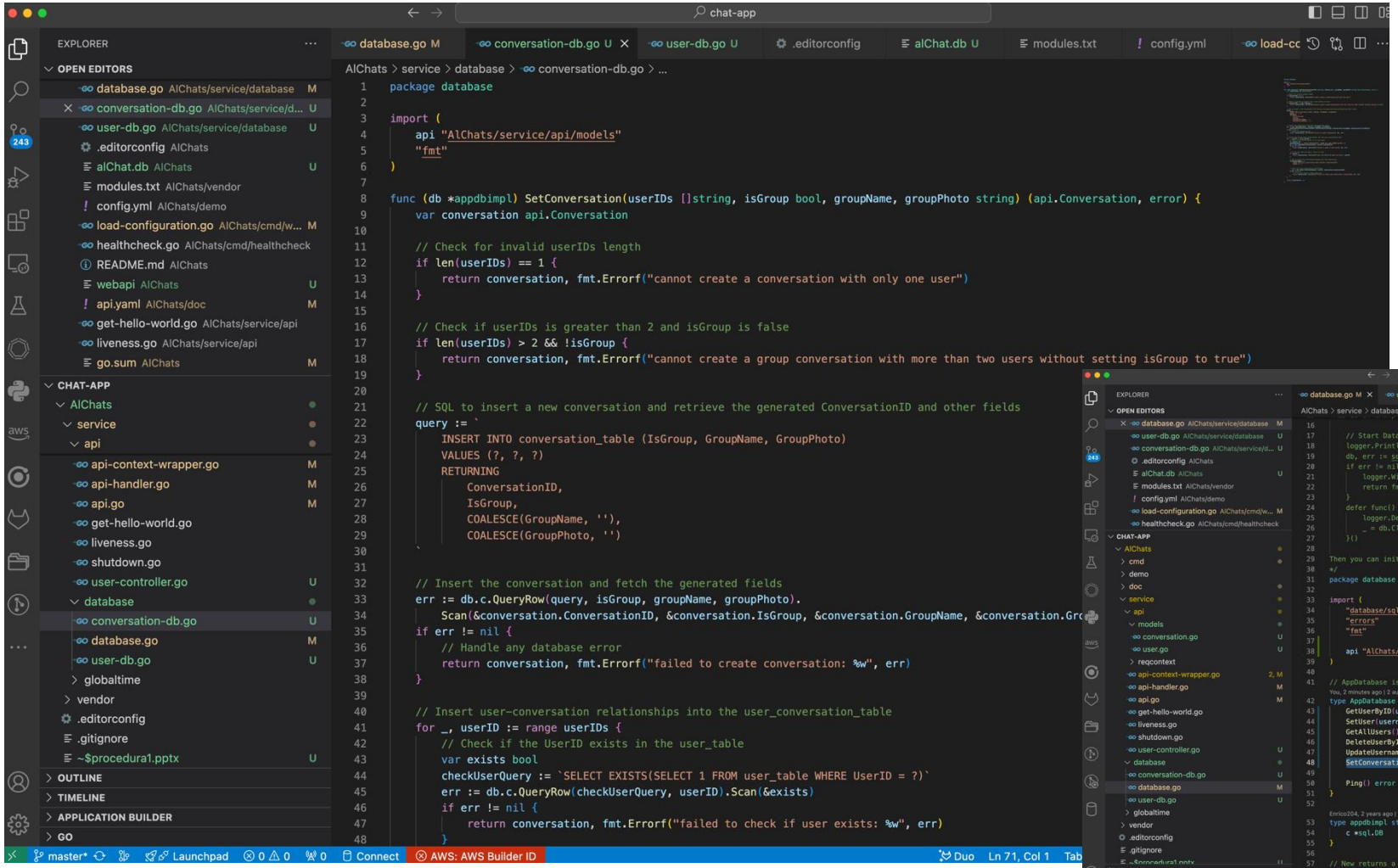
Creazione funzioni DB per Conversation

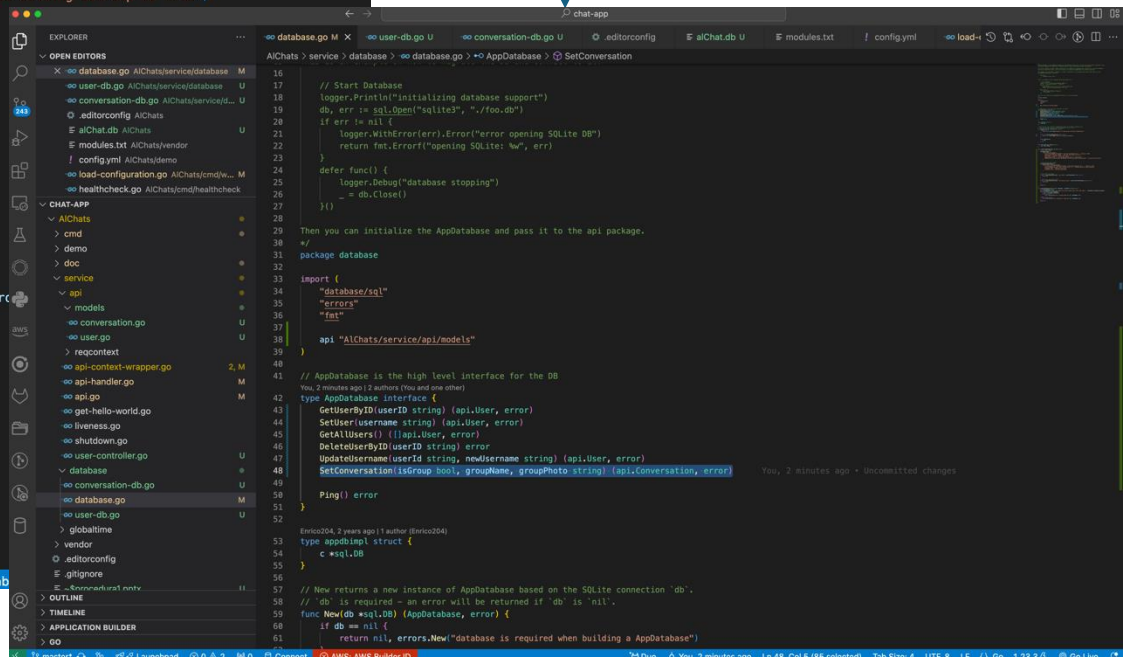


chatGPT

Creazione funzioni DB per Conversation

SetConversation(isGroup bool, groupName, groupPhoto string) (models.Conversation, error)





Creazione funzioni DB per Conversation

codice completo

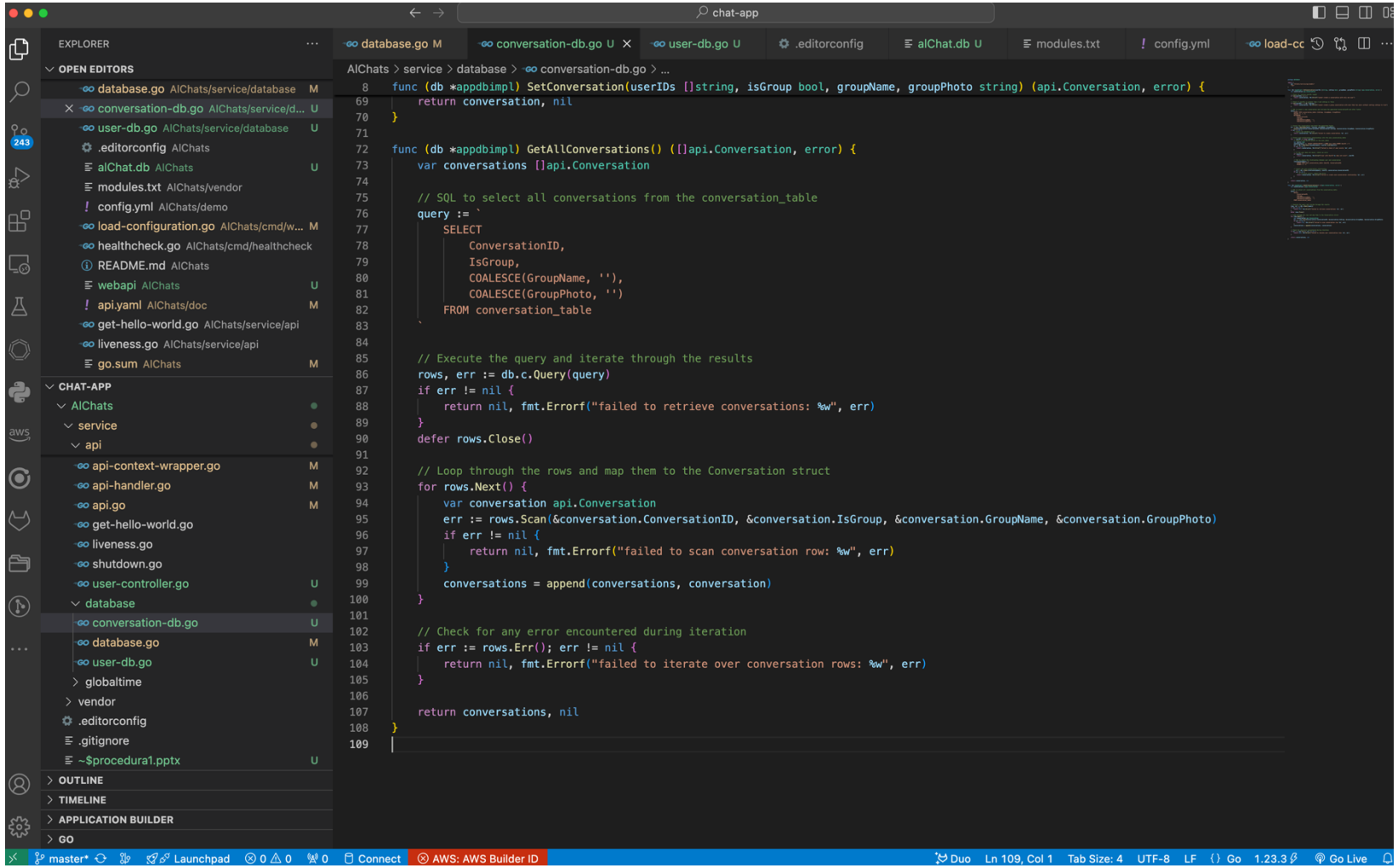
```
package database

import (
	api "AIChats/service/api/models"
	"fmt"
)

func (db *appdbimpl) SetConversation(userIDs []string, isGroup bool, groupName, groupPhoto string) (api.Conversation, error) {
	var conversation api.Conversation

	// Check for invalid userIDs length
	if len(userIDs) == 1 {
		return conversation, fmt.Errorf("cannot create a conversation with only one user")
	}

	// Check if userIDs is greater than 2 and isGroup is false
	if len(userIDs) > 2 && !isGroup {
		return conversation, fmt.Errorf("cannot create a group conversation with more than two users without setting isGroup to true")
	}

	// SQL to insert a new conversation and retrieve the generated ConversationID and other fields
	query := `
		INSERT INTO conversation_table (IsGroup, GroupName, GroupPhoto)
		VALUES (?, ?, ?)
		RETURNING
			ConversationID,
			IsGroup,
			COALESCE(GroupName, ''),
			COALESCE(GroupPhoto, '')
	`

	// Insert the conversation and fetch the generated fields
	err := db.c.QueryRow(query, isGroup, groupName, groupPhoto).
		Scan(&conversation.ConversationID, &conversation.IsGroup, &conversation.GroupName, &conversation.GroupPhoto)
	if err != nil {
		// Handle any database error
		return conversation, fmt.Errorf("failed to create conversation: %w", err)
	}

	// Insert user-conversation relationships into the user_conversation_table
	for _, userID := range userIDs {
		// Check if the UserID exists in the user_table
		var exists bool
		checkUserQuery := `SELECT EXISTS(SELECT 1 FROM user_table WHERE UserID = ?)`
		err := db.c.QueryRow(checkUserQuery, userID).Scan(&exists)
		if err != nil {
			return conversation, fmt.Errorf("failed to check if user exists: %w", err)
		}

		// If the user does not exist, return an error
		if !exists {
			return conversation, fmt.Errorf("user with UserID %s does not exist", userID)
		}

		// SQL to insert the relationship between user and conversation
		relationshipQuery := `
			INSERT INTO user_conversation_table (UserID, ConversationID)
			VALUES (?, ?)
		`

		// Insert the user-conversation relationship
		_, err = db.c.Exec(relationshipQuery, userID, conversation.ConversationID)
		if err != nil {
			// If an error occurs, return the error
			return conversation, fmt.Errorf("failed to create user-conversation relationship: %w", err)
		}
	}

	return conversation, nil
}
```

verifica se gli userid passati sono giusti (considerando group o coversazione

aggiorna tabella relazione molti-molti dopo aver verificato che ogni utente esiste

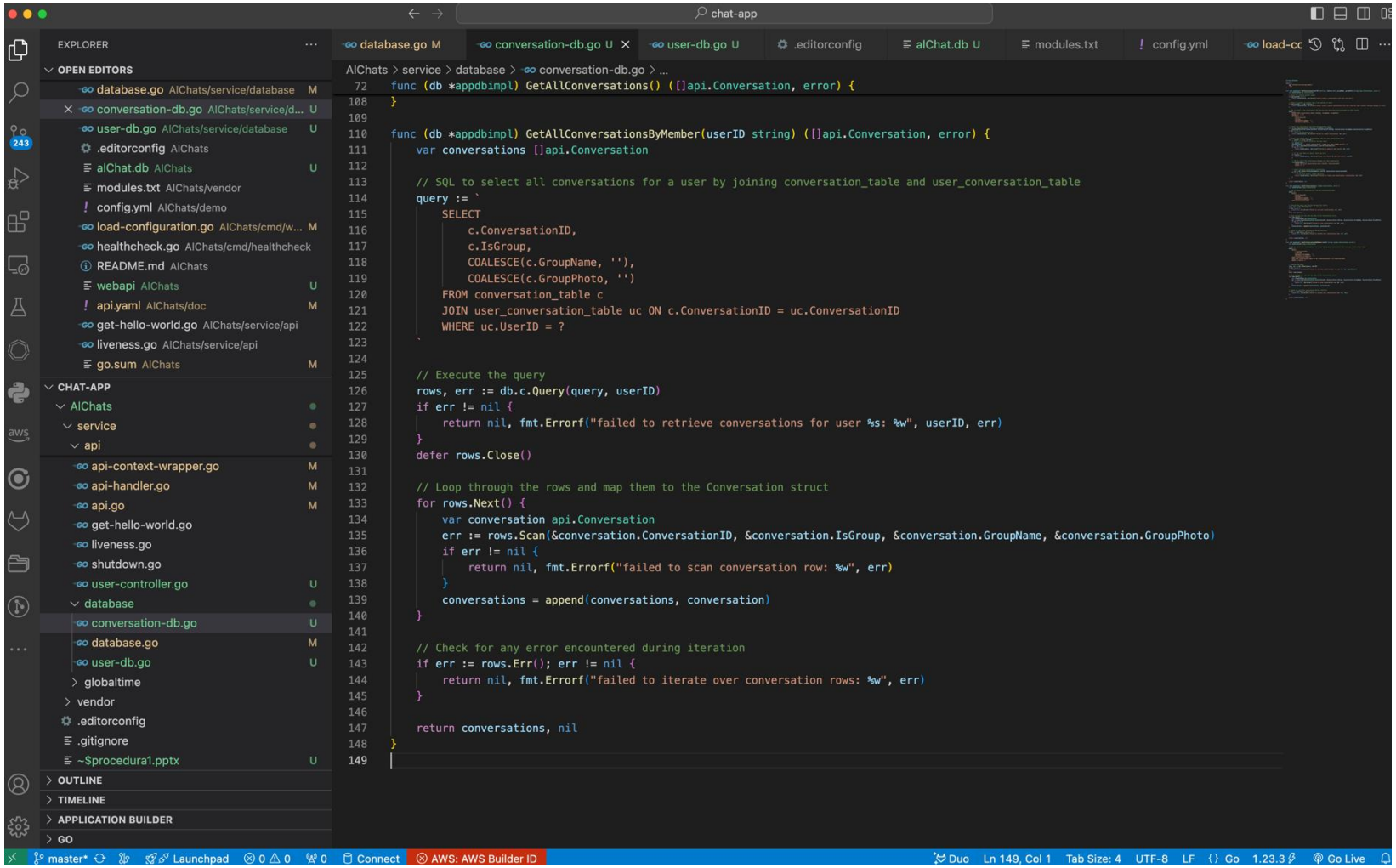
Creazione funzioni DB per Conversation

```
GetAllConversations() ([]api.Conversation, error)
```



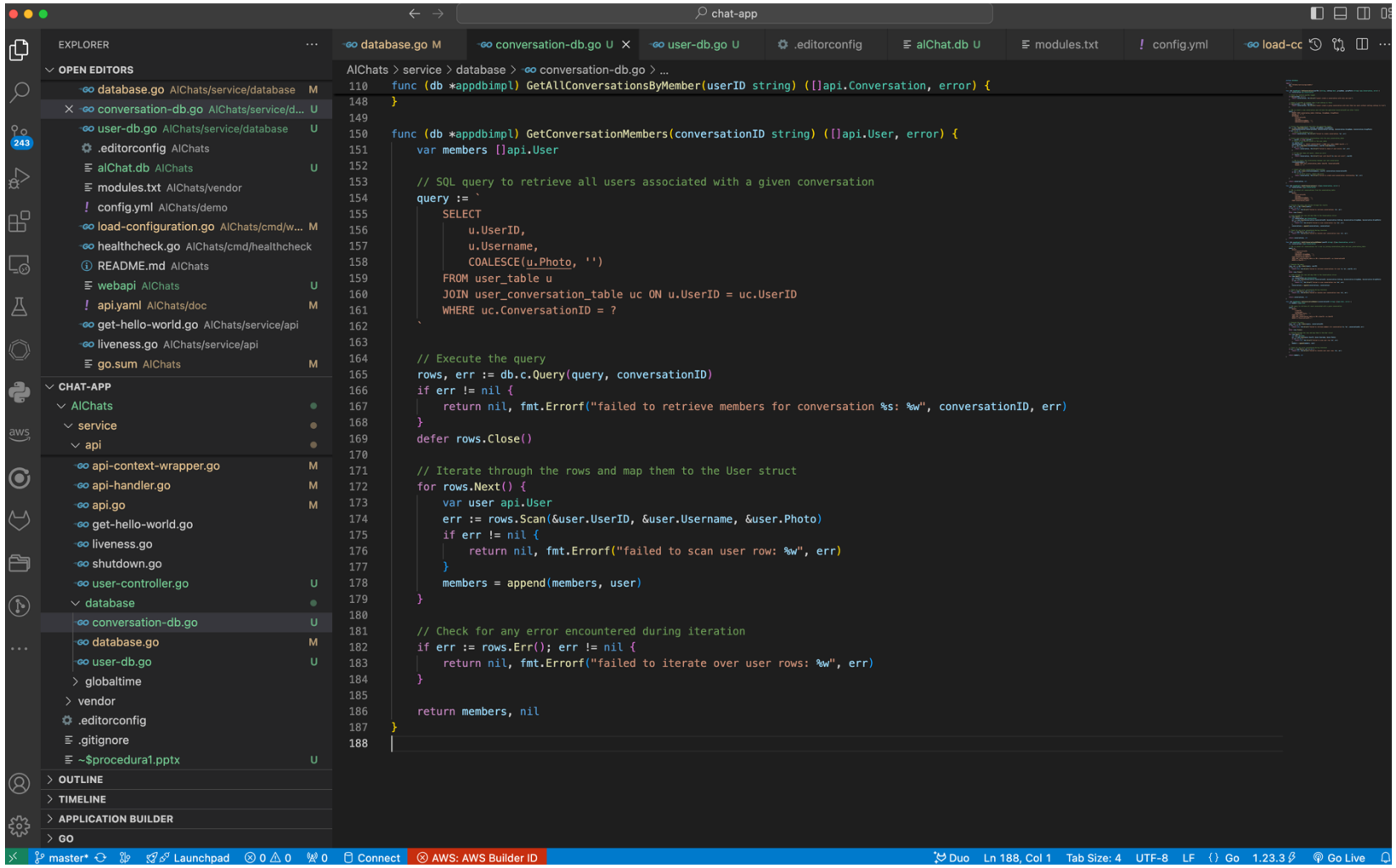
Creazione funzioni DB per Conversation

```
GetAllConversationsByMember(userID string) ([]api.Conversation, error)
```

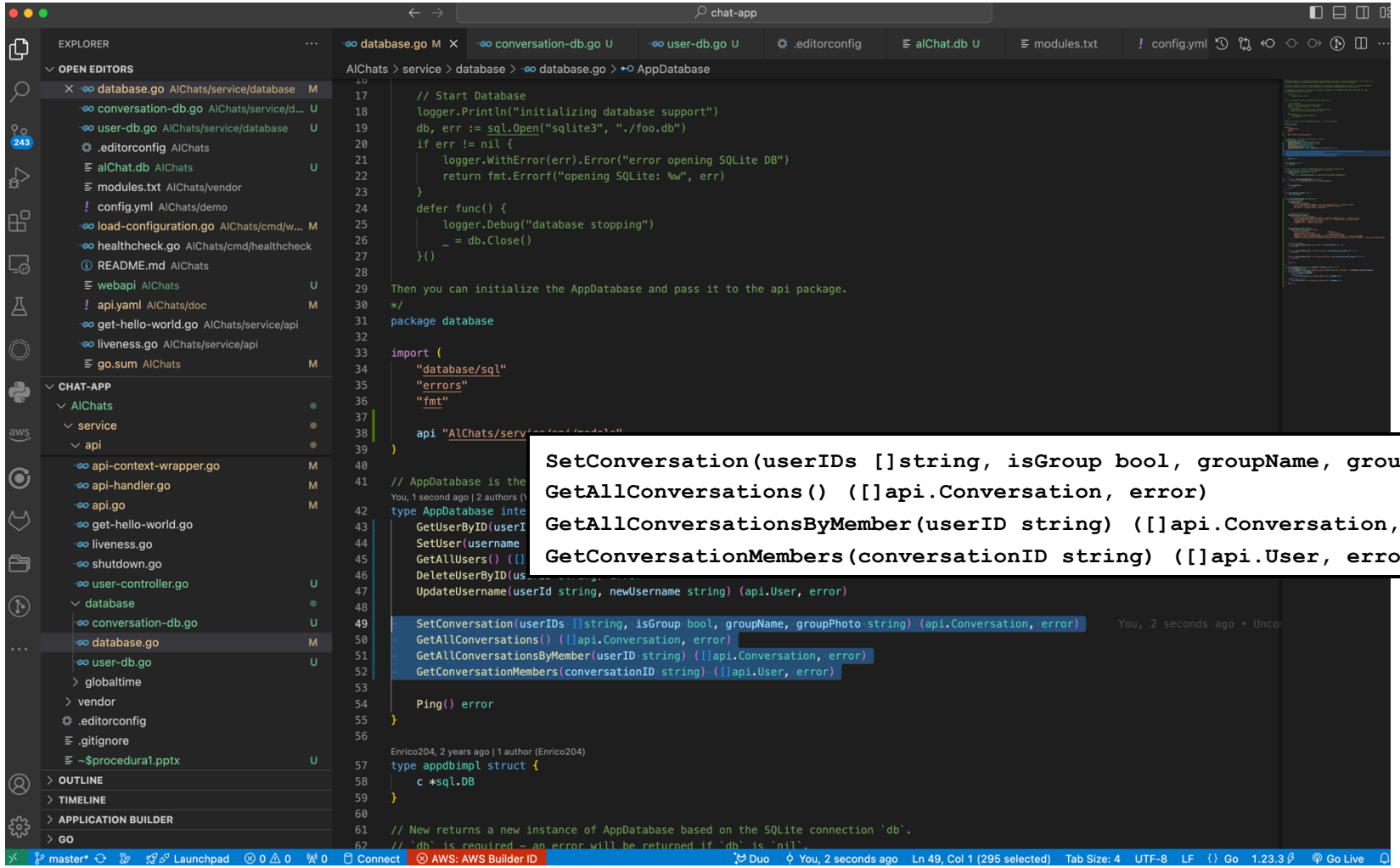


Creazione funzioni DB per Conversation

```
GetConversationMembers(conversationID string) ([]api.User, error)
```



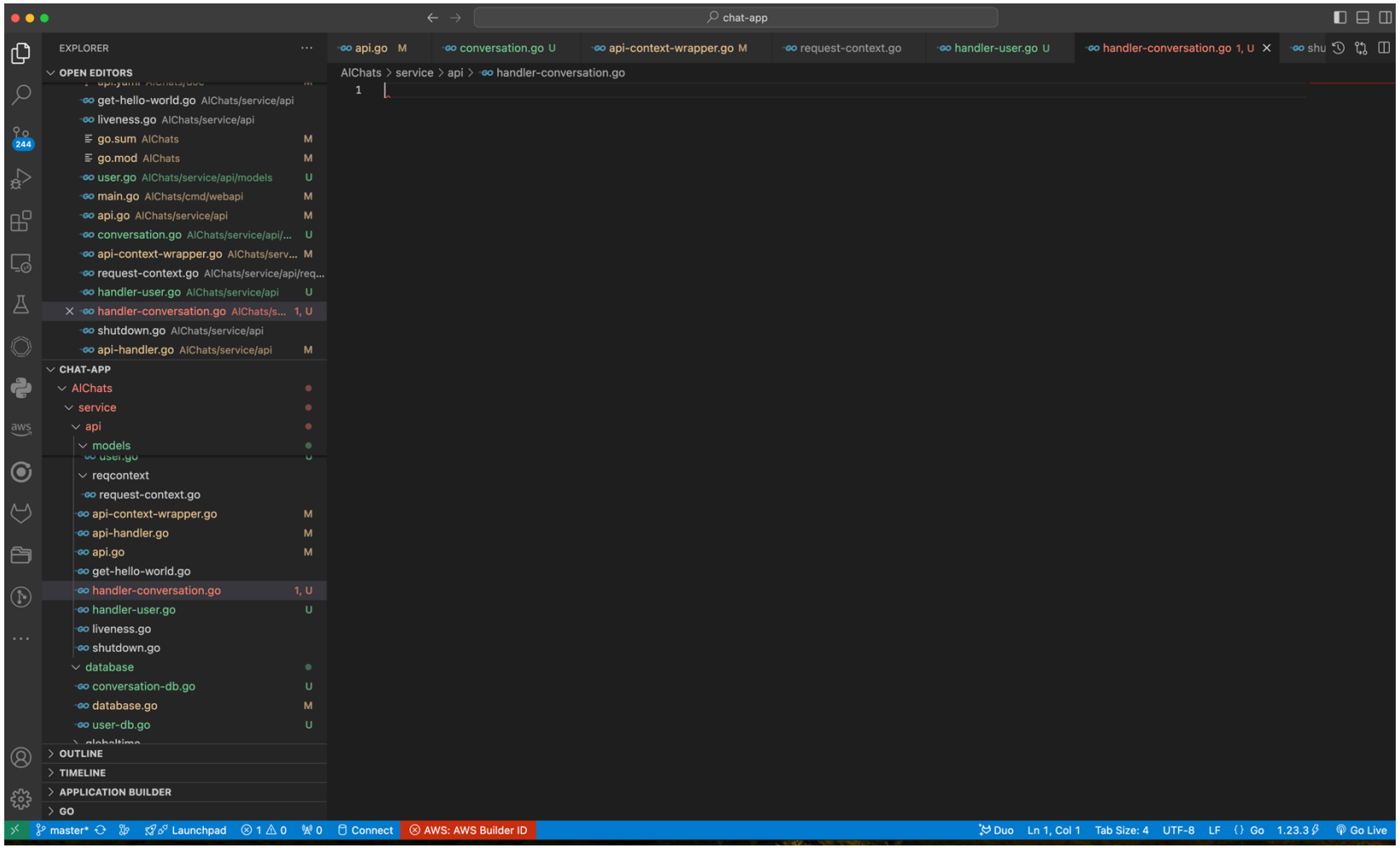
Creazione funzioni DB per Conversation




```
SetConversation(userIDs []string, isGroup bool, groupName, groupPhoto string) (api.Conversation, error)
GetAllConversations() ([]api.Conversation, error)
GetAllConversationsByMember(userID string) ([]api.Conversation, error)
GetConversationMembers(conversationID string) ([]api.User, error)
```

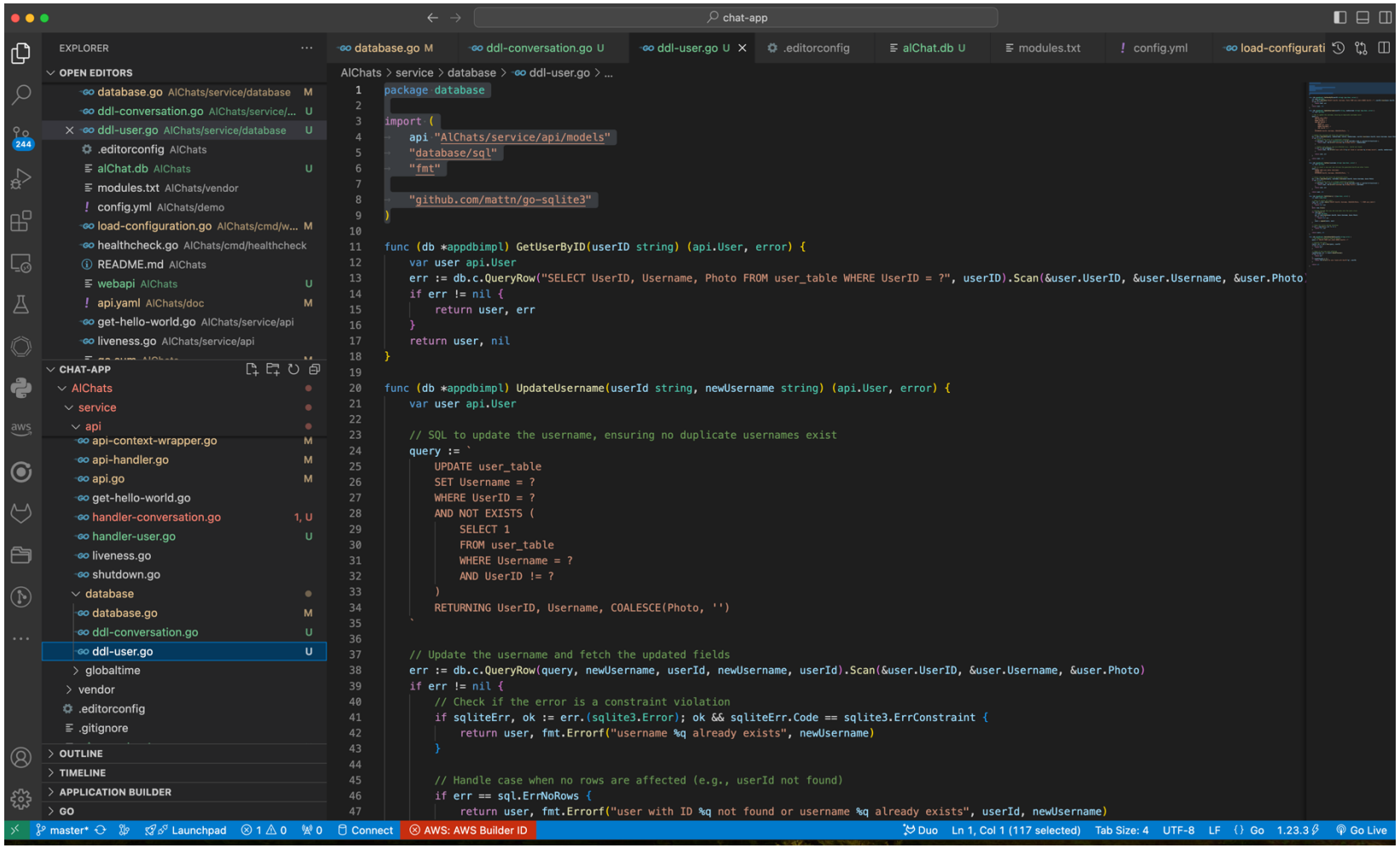
Creazione endpoint HTTP per Conversation

cambio nome dei controller in handler (handler-user ed il nuovo handler-conversation



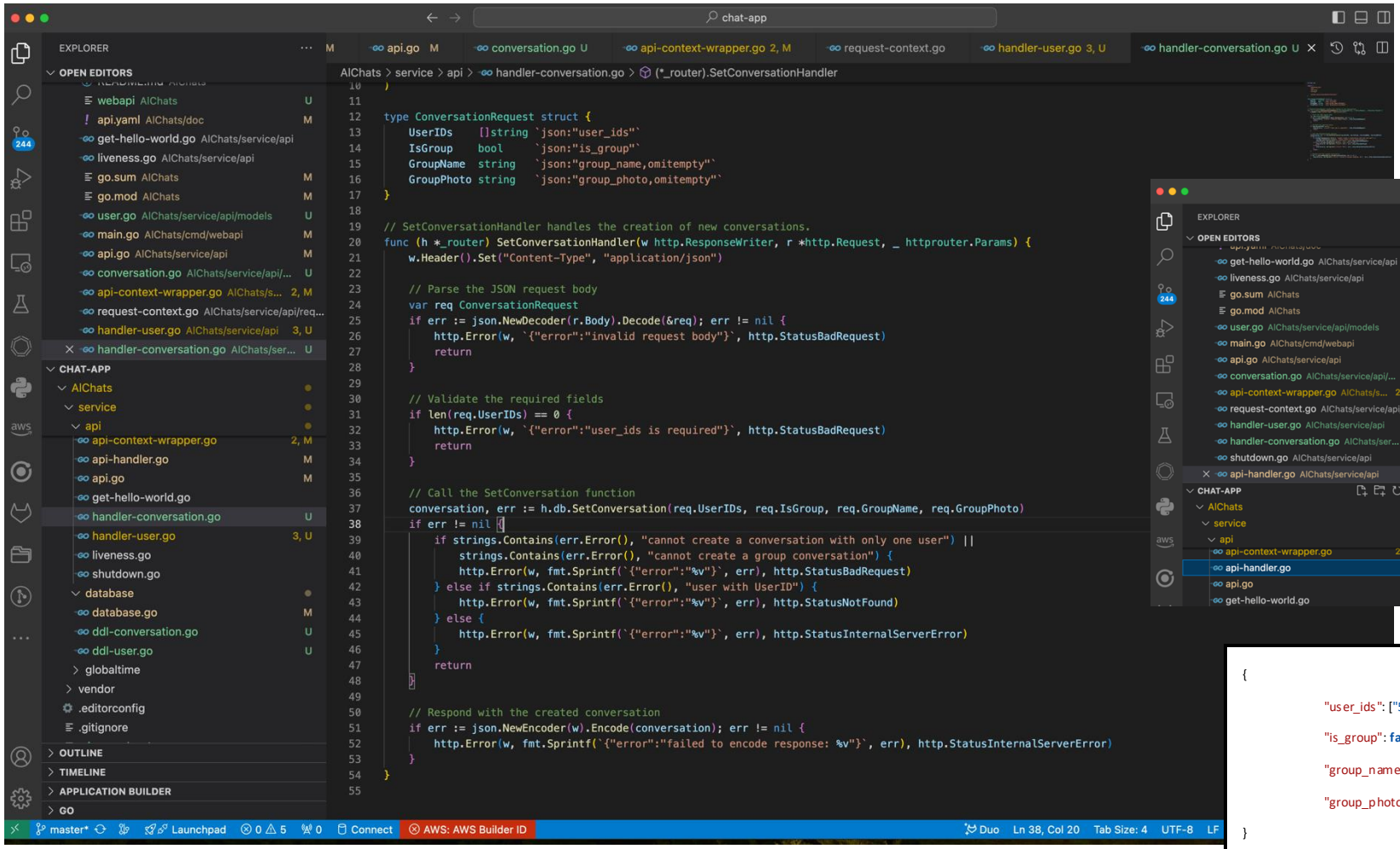
Creazione endpoint HTTP per Conversation

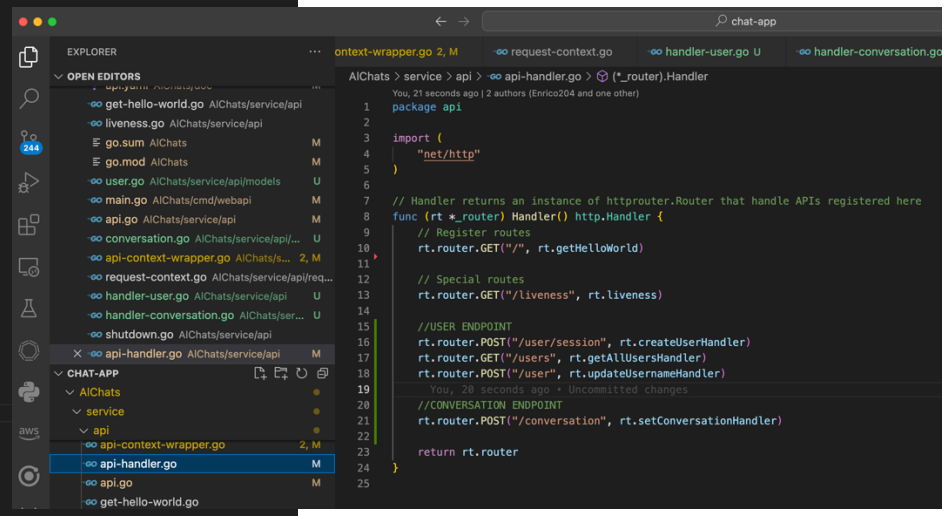
cambio nome anche ai file del db che contengono le query DDL (data definition language)
user-db.go diventa ddl.user.go (solo per una questione di leggibilità)



Creazione endpoint HTTP per Conversation

creo ConversationRequest che mi serve per definire il body della POST
creo il SetConversationhandler e lo dichiaro nel router






```
{
    "user_ids": ["5eef00fcbfddfa3848b9f850e1015184", "daf495832109b2785bb525055633c4c9"],
    "is_group": false,
    "group_name": "",
    "group_photo": "https://example.com/photo.png"
}
```

Creazione endpoint HTTP per Conversation

test endpoint

cancellazione db

```
pcarlo@DLJ473C2T5 chat-app % cd AlChats
pcarlo@DLJ473C2T5 AlChats % ls -al al*
-rw-r--r--  1 pcarlo  staff  16384  3 Dic 12:58 alChat.db
pcarlo@DLJ473C2T5 AlChats % rm alChat.db
pcarlo@DLJ473C2T5 AlChats % ls -al al*
zsh: no matches found: al*
pcarlo@DLJ473C2T5 AlChats %
```

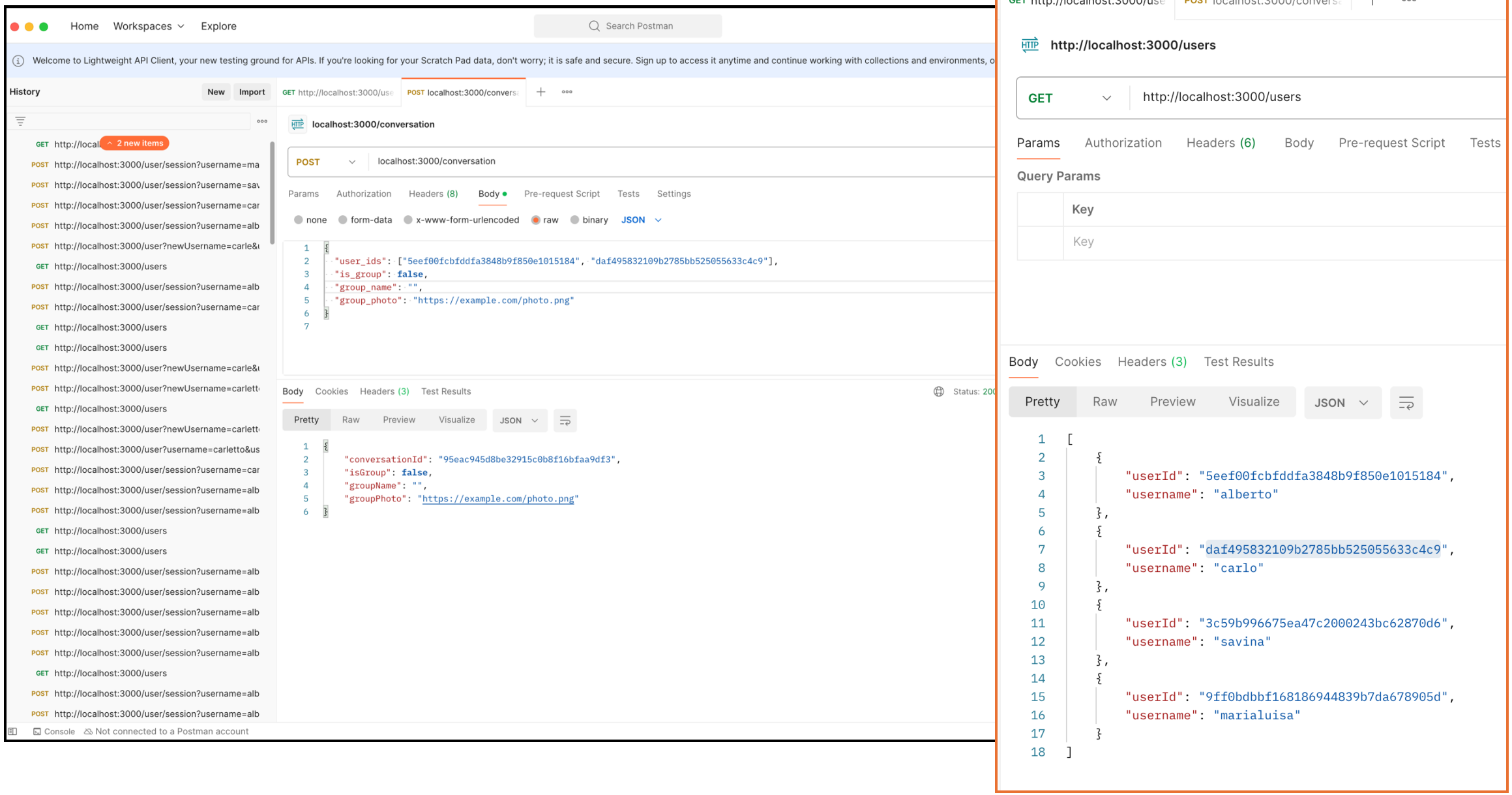
avvio programma

```
pcarlo@DLJ473C2T5 AlChats % go run ./cmd/webapi/
INFO[0000] application initializing
INFO[0000] initializing database support
INFO[0000] initializing API server
INFO[0000] API listening on 0.0.0.0:3000
```

check database

```
sqlite> SELECT name FROM sqlite_master WHERE type='table';
user_table
conversation_table
user_conversation_table
sqlite> select * from user_table;
sqlite> select * from conversation_table;
sqlite> select * from user_conversation_table;
sqlite>
```

nessun dato


Home
Workspaces
Explore
Search Postman
Welcome to Lightweight API Client, your new testing ground for APIs. If you're looking for your Scratch Pad data, don't worry; it is safe and secure. Sign up to access it anytime and continue working with collections and environments, o
History
New
Import
GET http://localhost:3000/use
POST localhost:3000/convers
localhost:3000/conversation
POST
localhost:3000/conversation
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
JSON
{
"user_ids": ["5eef00fcbfddfa3848b9f850e1015184", "daf495832109b2785bb525055633c4c9"],
"is_group": false,
"group_name": "",
"group_photo": "https://example.com/photo.png"
}
Body
Cookies
Headers (3)
Test Results
Status: 200
Pretty
Raw
Preview
Visualize
JSON
{
"conversationId": "95eac945d8be32915c0b8f16bfaa9df3",
"isGroup": false,
"groupName": "",
"groupPhoto": "https://example.com/photo.png"
}
GET http://localhost:3000/user?newUsername=carle&
GET http://localhost:3000/users
POST http://localhost:3000/user/session?username=alb
POST http://localhost:3000/user?newUsername=carlett
POST http://localhost:3000/user?username=carletto&us
Console
Not connected to a Postman account

GET http://localhost:3000/use
POST localhost:3000/convers
http://localhost:3000/users
GET
http://localhost:3000/users
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Query Params
Key
Key
Body
Cookies
Headers (3)
Test Results
Pretty
Raw
Preview
Visualize
JSON
[
{
"userId": "5eef00fcbfddfa3848b9f850e1015184",
"username": "alberto"
},
{
"userId": "daf495832109b2785bb525055633c4c9",
"username": "carlo"
},
{
"userId": "3c59b996675ea47c2000243bc62870d6",
"username": "savina"
},
{
"userId": "9ff0bdbbf168186944839b7da678905d",
"username": "marialuisa"
}
]

Creazione endpoint HTTP per Conversation

test endpoint

cancellazione db

```
pcarlo@DLJ473C2T5 chat-app % cd AlChats
pcarlo@DLJ473C2T5 AlChats % ls -al al*
-rw-r--r--  1 pcarlo  staff  16384  3 Dic 12:58 alChat.db
pcarlo@DLJ473C2T5 AlChats % rm alChat.db
pcarlo@DLJ473C2T5 AlChats % ls -al al*
zsh: no matches found: al*
pcarlo@DLJ473C2T5 AlChats %
```

avvio programma

```
pcarlo@DLJ473C2T5 AlChats % go run ./cmd/webapi/
INFO[0000] application initializing
INFO[0000] initializing database support
INFO[0000] initializing API server
INFO[0000] API listening on 0.0.0.0:3000
```

check database

```
sqlite> select * from user_table;
5eef00fcbfddfa3848b9f850e1015184|alberto|
daf495832109b2785bb525055633c4c9|carlo|
3c59b996675ea47c2000243bc62870d6|savina|
9ff0bdbbf168186944839b7da678905d|marialuisa|
sqlite> select * from conversation_table;
95eac945d8be32915c0b8f16bfaa9df3|0||https://example.com/photo.png
sqlite> select * from user_conversation_table;
5eef00fcbfddfa3848b9f850e1015184|95eac945d8be32915c0b8f16bfaa9df3
daf495832109b2785bb525055633c4c9|95eac945d8be32915c0b8f16bfaa9df3
sqlite>
```