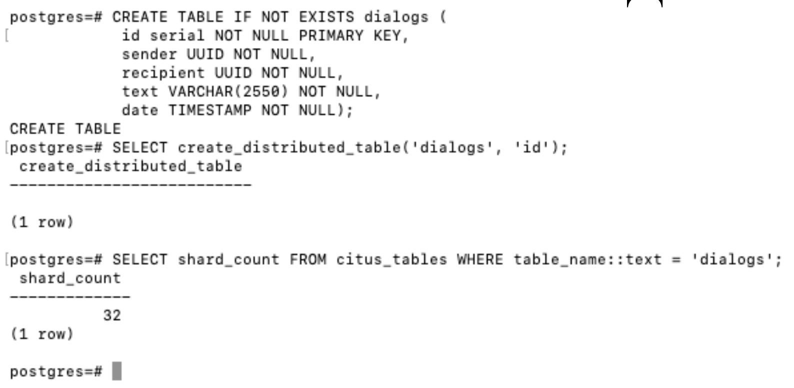
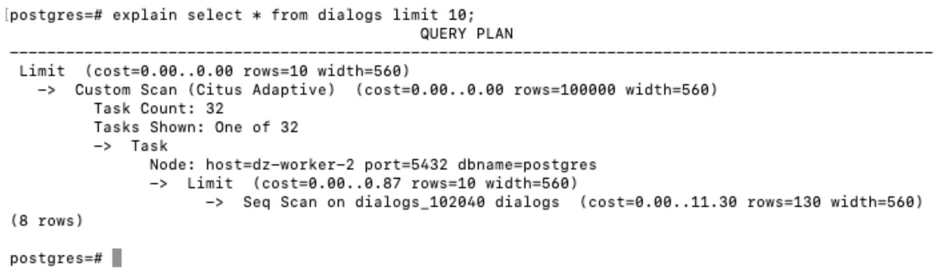
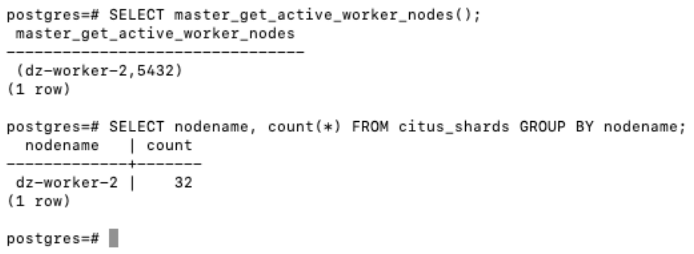
# Отчет ДЗ №7



```
postgres=# CREATE TABLE IF NOT EXISTS dialogs (
            id serial NOT NULL PRIMARY KEY,
            sender UUID NOT NULL,
            recipient UUID NOT NULL,
            text VARCHAR(2550) NOT NULL,
            date TIMESTAMP NOT NULL);
CREATE TABLE
postgres=# SELECT create_distributed_table('dialogs', 'id');
 create_distributed_table
--------------------------

(1 row)

postgres=# SELECT shard_count FROM citus_tables WHERE table_name::text = 'dialogs';
 shard_count
-------------
          32
(1 row)

postgres=# 
```

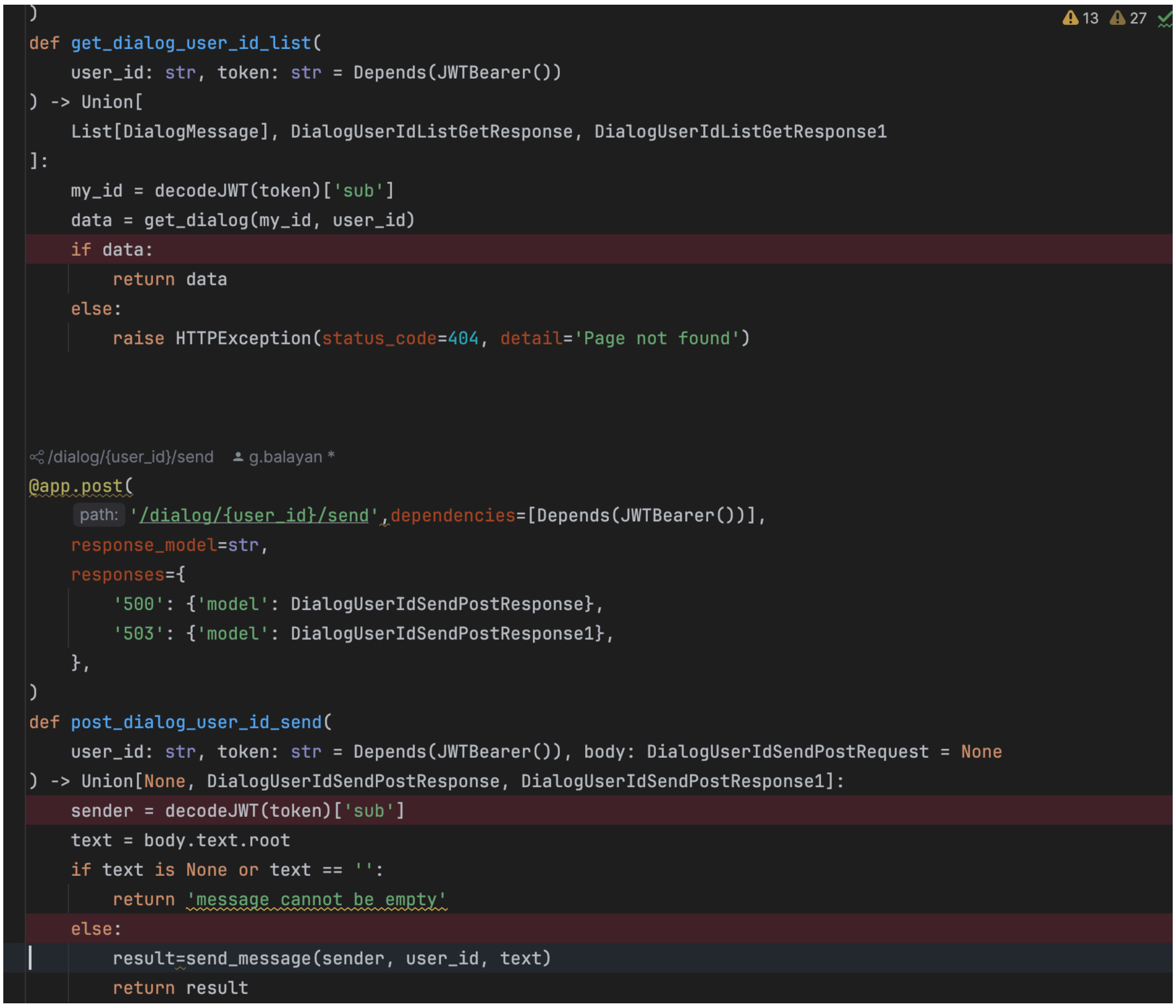


```
postgres=# explain select * from dialogs limit 10;
                                              QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Limit  (cost=0.00..0.00 rows=10 width=560)
   ->  Custom Scan (Citus Adaptive)  (cost=0.00..0.00 rows=100000 width=560)
         Task Count: 32
         Tasks Shown: One of 32
         ->  Task
               Node: host=dz-worker-2 port=5432 dbname=postgres
               ->  Limit  (cost=0.00..0.87 rows=10 width=560)
                     ->  Seq Scan on dialogs_102040 dialogs  (cost=0.00..11.30 rows=130 width=560)
(8 rows)

postgres=# 
```



```
postgres=# SELECT master_get_active_worker_nodes();
 master_get_active_worker_nodes
--------------------------------
 (dz-worker-2,5432)
(1 row)

postgres=# SELECT nodename, count(*) FROM citus_shards GROUP BY nodename;
  nodename   | count
-------------+-------
 dz-worker-2 |    32
(1 row)

postgres=# 
```

Создал шардированную таблицу в citus. Добавил в database.ini еще один тип подключения(подключение к citus)



```
)
def get_dialog_user_id_list(
    user_id: str, token: str = Depends(JWTBearer())
) -> Union[
    List[DialogMessage], DialogUserIdListGetResponse, DialogUserIdListGetResponse1
]:
    my_id = decodeJWT(token)['sub']
    data = get_dialog(my_id, user_id)
    if data:
        return data
    else:
        raise HTTPException(status_code=404, detail='Page not found')



/dialog/{user_id}/send   g.balayan *
@app.post(
    path: '/dialog/{user_id}/send',dependencies=[Depends(JWTBearer())],
    response_model=str,
    responses={
        '500': {'model': DialogUserIdSendPostResponse},
        '503': {'model': DialogUserIdSendPostResponse1},
    },
)
def post_dialog_user_id_send(
    user_id: str, token: str = Depends(JWTBearer()), body: DialogUserIdSendPostRequest = None
) -> Union[None, DialogUserIdSendPostResponse, DialogUserIdSendPostResponse1]:
    sender = decodeJWT(token)['sub']
    text = body.text.root
    if text is None or text == '':
        return 'message cannot be empty'
    else:
        result=send_message(sender, user_id, text)
        return result
```

Реализовал ручки отправки и получения сообщений.