**Δεληγιάννη Μυρτώ 1067389 – Τμήμα Μηχανικών Η/Υ και Πληροφορικής**

**Φεβρουάριος 2024**

**Κατανεμημένα Συστήματα – Project**

**2.2.1 Μοντέλο Αποστολής – Παράδοσης Μηνυμάτων μέσω Κοινής Ουράς**

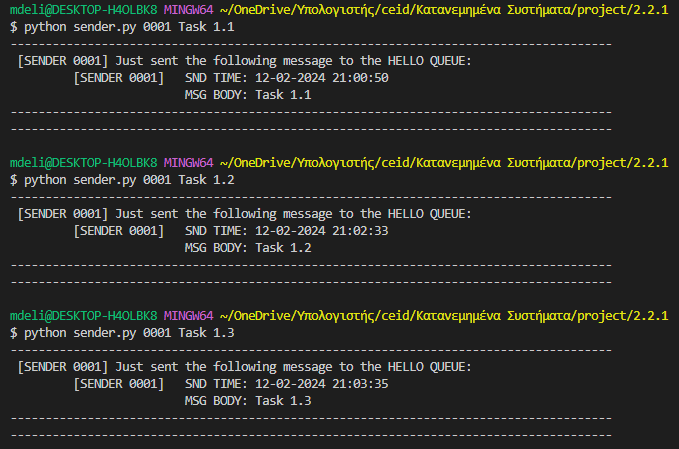
Αξιοποιώντας τον σύνδεσμο <https://www.rabbitmq.com/tutorials/tutorial-two-python.html>, προσθέτοντας τις απαραίτητες παραμέτρους (αναγνωριστικά αποστολέα, παραλήπτη και μηνύματος/διεργασίας, χρόνοι ενεργοποίησης, αποστολής και παραλαβής των μηνυμάτων/διεργασιών), υλοποιήθηκε μια απλή υπηρεσία στην οποία, διάφοροι παραλήπτες λαμβάνουν ασύγχρονα μηνύματα από αυθαίρετους αποστολείς και από τον message broker ακολουθείται πολιτική Round Robin.

Δυο παραδείγματα εκτέλεσης:

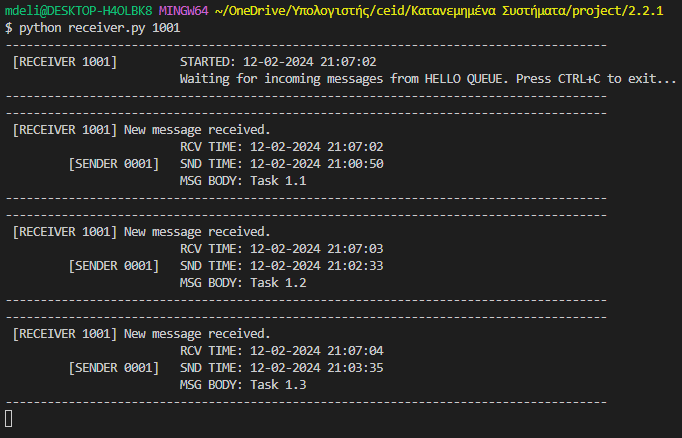
1. Ένας αποστολέας ενεργοποιείται και στέλνει μηνύματα. Αργότερα, μόλις

ενεργοποιηθεί ο παραλήπτης, ξεκινάει να εξυπηρετεί με τη σειρά τα μηνύματα.

SENDER:

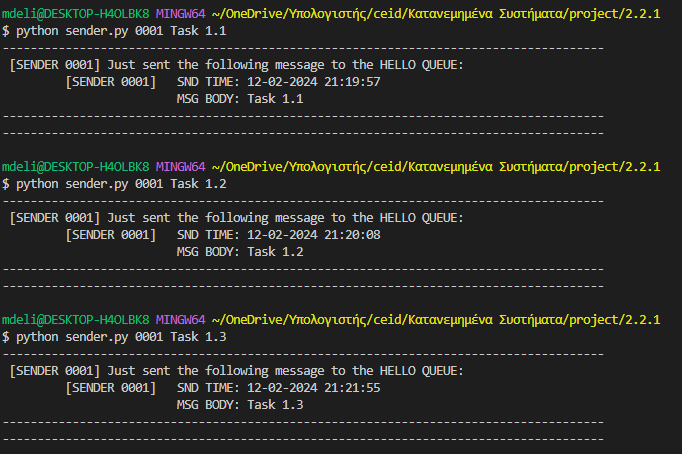


RECEIVER:

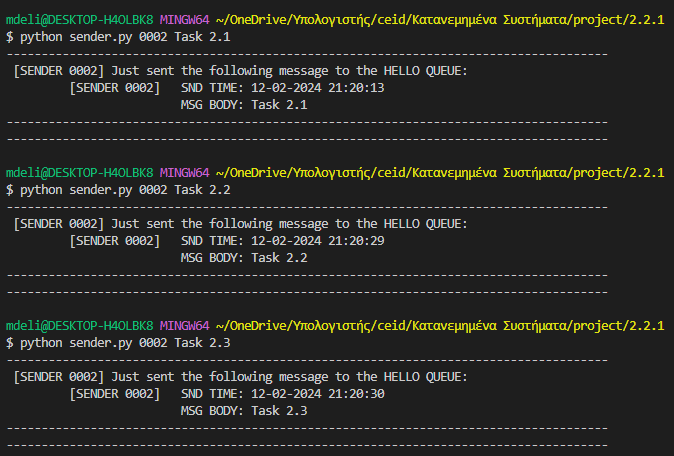


1. Πρώτα ενεργοποιούνται 3 παραλήπτες και περιμένουν, ύστερα ενεργοποιούνται 2 αποστολείς και στέλνουν από 3 μηνύματα ο καθένας

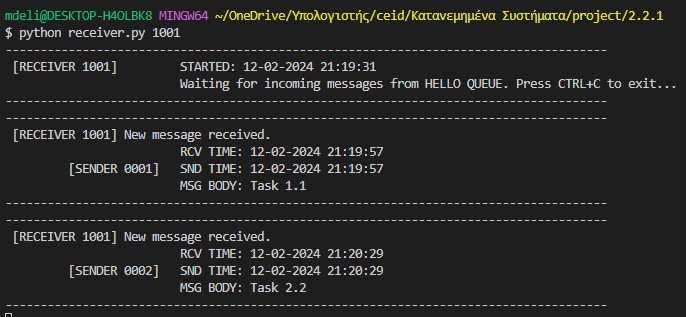
SENDER 1:



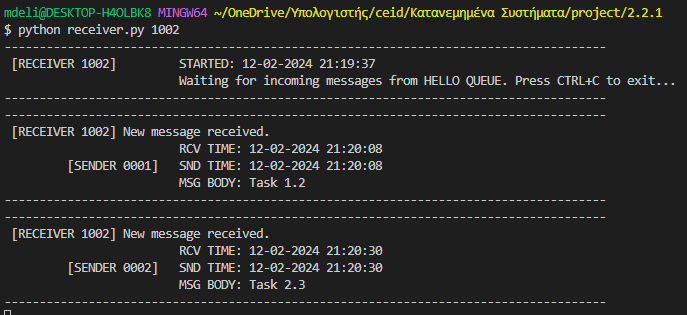
SENDER 2:



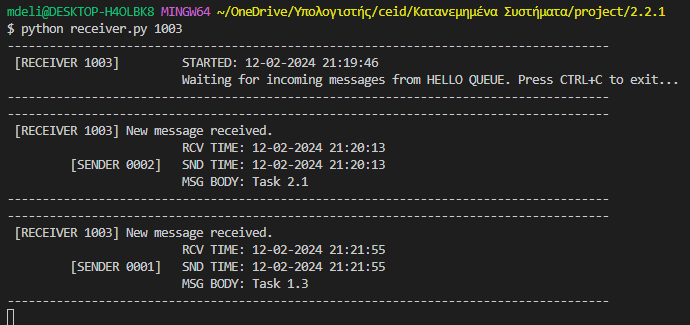
RECEIVER 1:



RECEIVER 2:



RECEIVER 3:

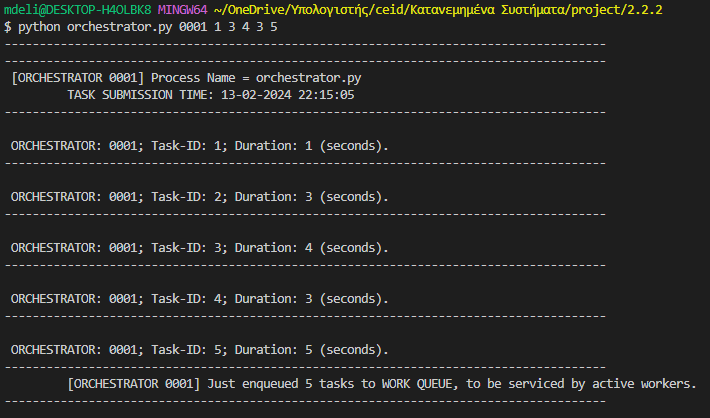


* + 1. **Στοιχειώδες Μοντέλο Ενορχήστρωσης Ανάθεσης Καθηκόντων**

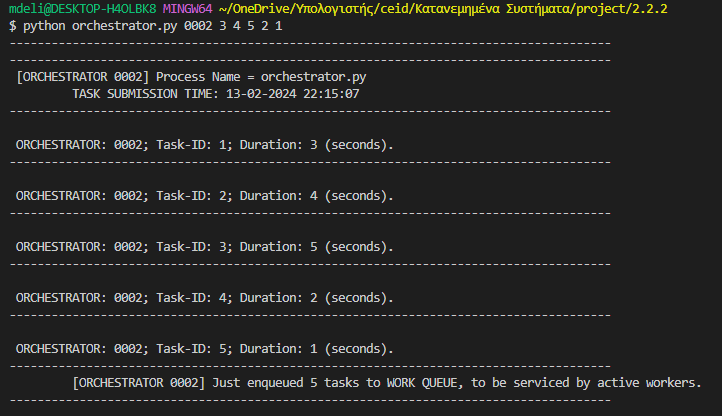
Τροποποιώντας τον κώδικα του προηγούμενου ερωτήματος ώστε ο ενορχηστρωτής (πρώην-αποστολέας) να αποστέλλει στους εργάτες μια ουρά διεργασιών αντί για μια μοναδική εργασία τη φορά, επιτυγχάνεται το Στοιχειώδες Μοντέλο Ενορχήστρωσης Ανάθεσης Καθηκόντων. Κάθε ξεχωριστή διεργασία στην ουρά, έχει τον δικό της εκτιμώμενο χρόνο εκτέλεσης, τον οποίο λαμβάνει υπόψιν ο εκάστοτε εργάτης που θα την εκτελέσει. Αν κάποιος εργάτης ξαφνικά σταματήσει να είναι ενεργός, η διεργασία δεν χάνεται. Θα μεταφερθεί στην ουρά «νεκρών» μηνυμάτων (dead – letter queue) και αργότερα θα επιστρέψει πίσω στην αρχική ουρά για να αποσταλλεί ξανά σε κάποιον από τους εναπομείναντες ενεργούς εργάτες.

Τα παραπάνω φαίνονται στο επόμενο παράδειγμα. Αρχικά δημιουργούμε 3 εργάτες που περιμένουν να εξυπηρετήσουν. Έπειτα δημιουργούμε 2 ενορχηστρωτές, οι οποίοι αποστέλλουν από 5 διεργασίες διαφορετικής χρονικής διάρκειας ο καθένας. Κάποια στιγμή παρατηρούμε ότι η λειτουργία του τελευταίου εργάτη (με ID: 9003) διακόπτεται κατά την εκτέλεση της διεργασίας με ID: 4 που εστάλη από τον ενορχηστρωτή με ID: 0001. Ωστόσο, μπορούμε να δούμε ότι η συγκεκριμένη εργασία δεν χάθηκε, αφού αργότερα εκτελέστηκε επιτυχώς από τον εργάτη με ID: 9001.

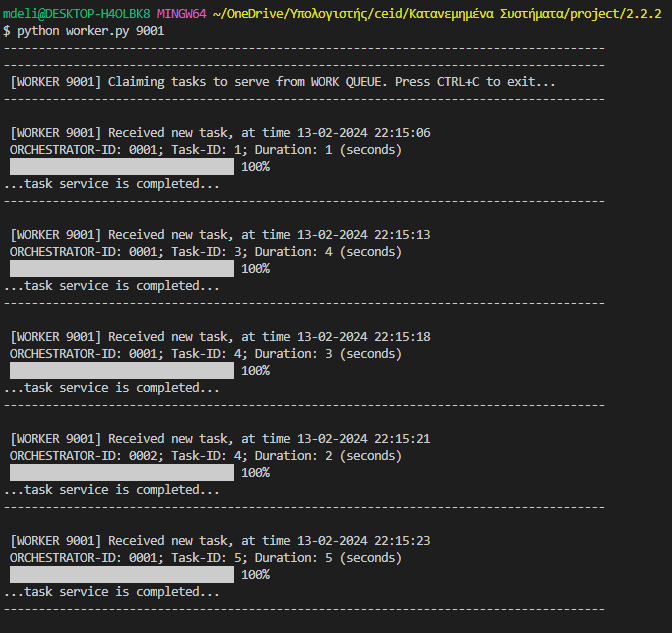
ORCHESTRATOR 0001:



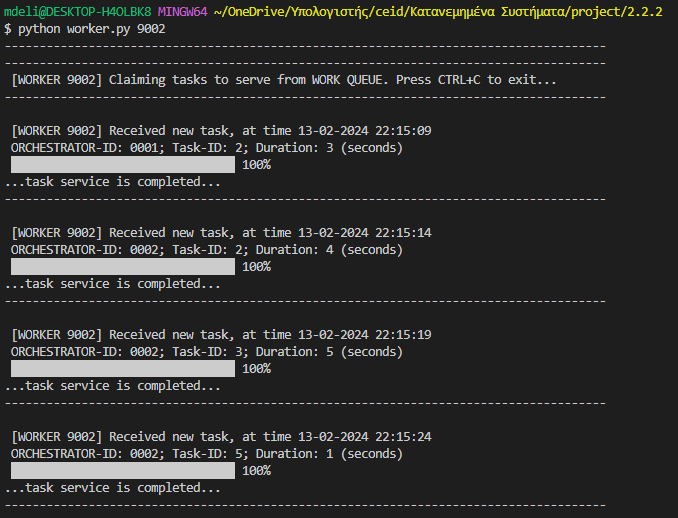
ORCHESTRATOR 0002:



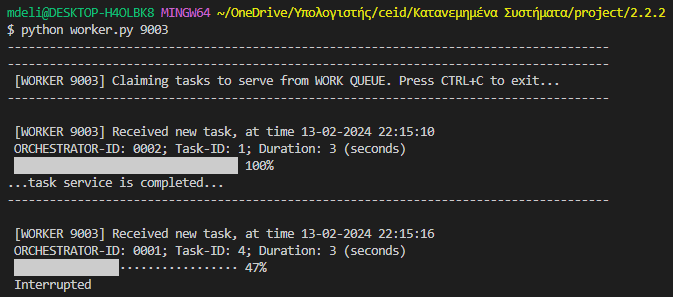
WORKER 9001:



WORKER 9002:



WORKER 9003:



**2.2.3 Μοντέλο Φιλτραρίσματος Μηνυμάτων**

Σε αυτή τη φάση, αξιοποιώντας την έννοια του ανταλλακτηρίου (exchange), οι προηγούμενοι κώδικες τροποιποιήθηκαν με βάση τα παραδείγματα από το rabbitmq.com ώστε να υιοθετηθεί το ζητούμενο μοτίβο επικοινωνίας. Κάθε publisher, είτε είναι ενορχηστρωτής, είτε εργάτης, αποστέλλει ένα μήνυμα με ένα tag, μια ετικέτα, που δηλώνει σε ποιά ομάδα ή σε ποιόν subscriber συγκεκριμένα απευθύνεται το μήνυμα αυτό. Αυτό εγγράφεται στο ανταλλακτήριο task\_stream και περιμένει να εξυπηρετηθεί. Κάθε subscriber διαθέτει μια λίστα ετικετών που δηλώνουν ποιά μηνύματα μπορεί να εξυπηρετήσει. Αν σε κάποιο μήνυμα αναγνωρίσει κάποια από τις ετικέτες της λίστας, σημαίνει ότι μπορεί να το αναλάβει, αλλιώς το απορρίπτει και το μήνυμα συνεχίζει να περιμένει στην ουρά.

Στο επόμενο παράδειγμα εκτελούνται οι εντολές της εκφώνησης για να επιβεβαιωθεί η παραπάνω λειτουργία.

ORCH1-subscriber:

A screenshot of a computer

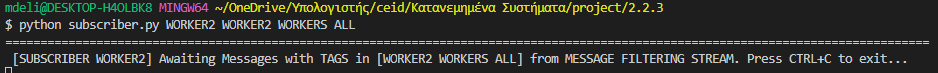
Description automatically generated

WORKER1-subscriber:

A screenshot of a computer

Description automatically generated

WORKER2-subscriber:



ORCH1-publisher:

A screenshot of a computer

Description automatically generated

WORKER1-publisher:

A screenshot of a computer

Description automatically generated

WORKER2-publisher:

A screenshot of a computer

Description automatically generated