

Μερικές σημειώσεις διευκρινίσεις για το πρότζεκτ:

Το πρότζεκτ έγινε σε spring boot.

Έχει χωριστεί σε 4 βασικά folders:

- 1) **Model:** εκεί βρίσκονται οι κλάσεις που περιγράφουν τις δομές δεδομένων:
 - i) **Account.java:** Περιέχει την κλάση Account, η οποία ορίζει ένα αντικείμενο λογαριασμού με accountId και beneficiaryId.
 - ii) **Beneficiary.java:** Κλάση που περιγράφει τον δικαιούχο με πεδία beneficiaryId, firstName, lastName.
 - iii) **Transaction.java:** Κλάση που αναπαριστά μια συναλλαγή με ιδιότητες transactionId, accountId, amount, type, date.
 - iv) **AccountBalance.java:** Κλάση που αναπαριστά το υπόλοιπο ενός λογαριασμού, με πεδία accountId και balance.
 - v) **AccountWithBalances.java:** Κλάση που συνδυάζει έναν Beneficiary με όλα τα AccountBalance που του ανήκουν.
 - vi) **BeneficiaryWithAccounts.java:** Κλάση που επιστρέφει έναν Beneficiary μαζί με όλα τα accounts που του ανήκουν.
 - vii) **BeneficiaryWithTransactions.java:** Κλάση που επιστρέφει έναν Beneficiary με όλες τις συναλλαγές από όλους τους λογαριασμούς του.
 - viii) **Exercise4RequestBody.java:** Κλάση που χρησιμοποιείται στην άσκηση 4 για την είσοδο δεδομένων (όνομα και επώνυμο χρήστη).
- 2) **CSVLoaders:** Ο φάκελος αυτός περιέχει τις κλάσεις που είναι υπεύθυνες για τη φόρτωση δεδομένων από τα αρχεία CSV:
 - i) **Accounts.java:** Περιέχει μέθοδο που επιστρέφει λίστα με όλα τα Account objects, ταξινομημένα με βάση το beneficiaryId.
 - ii) **Beneficiaries.java:** Περιέχει μέθοδο που επιστρέφει ArrayList με όλους τους Beneficiary objects.
 - iii) **Transactions.java:** Περιέχει μέθοδο που επιστρέφει ArrayList με όλα τα Transaction objects, ταξινομημένα με βάση το accountId.

Η φόρτωση των δεδομένων από τα CSV αρχεία υλοποιήθηκε μέσα σε **static block**. Με αυτόν τον τρόπο η διαδικασία εκτελείται μόνο μία φορά κατά την εκκίνηση της εφαρμογής και όχι κάθε φορά που ζητείται η λίστα δεδομένων, εξασφαλίζοντας καλύτερη απόδοση.

- 3) **Controlllers:** Σε αυτόν τον φάκελο βρίσκονται βοηθητικές κλάσεις/μέθοδοι που χρησιμοποιούνται στα endpoints, με σκοπό την απλοποίηση και επαναχρησιμοποίηση της λογικής:
 - i) **BinarySearchAccounts.java:** Εφόσον τα Account objects είναι ήδη ταξινομημένα με βάση το beneficiaryId, υλοποιείται μια παραλλαγή δυαδικής αναζήτησης. Με αυτήν τον τρόπο μπορούμε να εντοπίζουμε αποδοτικά ποια accounts ανήκουν σε έναν συγκεκριμένο δικαιούχο, αντί να γίνεται γραμμική αναζήτηση.
 - ii) **BinarySearchTransaction.java:** Εφόσον τα Transaction objects είναι ταξινομημένα με βάση το accountId, υλοποιείται αντίστοιχη παραλλαγή δυαδικής αναζήτησης. Έτσι

μπορούμε, δεδομένης μιας λίστας με account IDs, να βρούμε αποδοτικά όλα τα transactions που αντιστοιχούν σε κάθε account.

- iii) **BalanceCalculator.java**: Περιέχει μέθοδο που δέχεται μια λίστα από Transaction objects (από διάφορους λογαριασμούς) και επιστρέφει μια λίστα από AccountBalance objects, δηλαδή για κάθε λογαριασμό υπολογίζει και επιστρέφει το balance του.

- 4) Routes: Σε αυτόν τον φάκελο βρίσκονται οι REST διεπαφές (endpoints) της εφαρμογής. Κάθε άσκηση αντιστοιχεί σε συγκεκριμένο endpoint που υλοποιεί το ζητούμενο του project.

ΖΗΤΟΥΜΕΝΑ ΤΟΥ ΠΡΟΤΥΠΟΥ:

1)Ανάκτηση στοιχείων δικαιούχου.

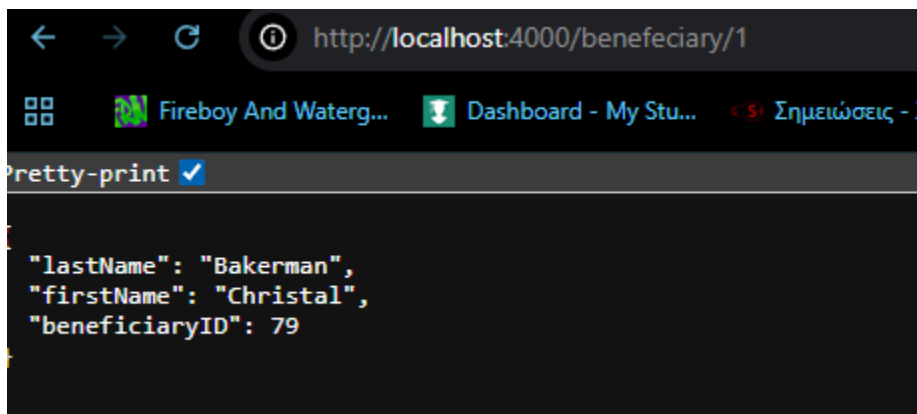
Exercise1.java

Το endpoint δέχεται ως **path parameter** το accountId του δικαιούχου

Αναζητούμε στη λίστα με τα accounts το Account object που έχει το συγκεκριμένο accountId.

Από το αντικείμενο αυτό παίρνουμε το beneficiaryId.

Με βάση το beneficiaryId, αναζητούμε στη λίστα με τους beneficiaries και επιστρέφουμε τα στοιχεία του δικαιούχου (π.χ. όνομα, επώνυμο, beneficiary ID).



2. Ανάκτηση των λογαριασμών ενός δικαιούχου.

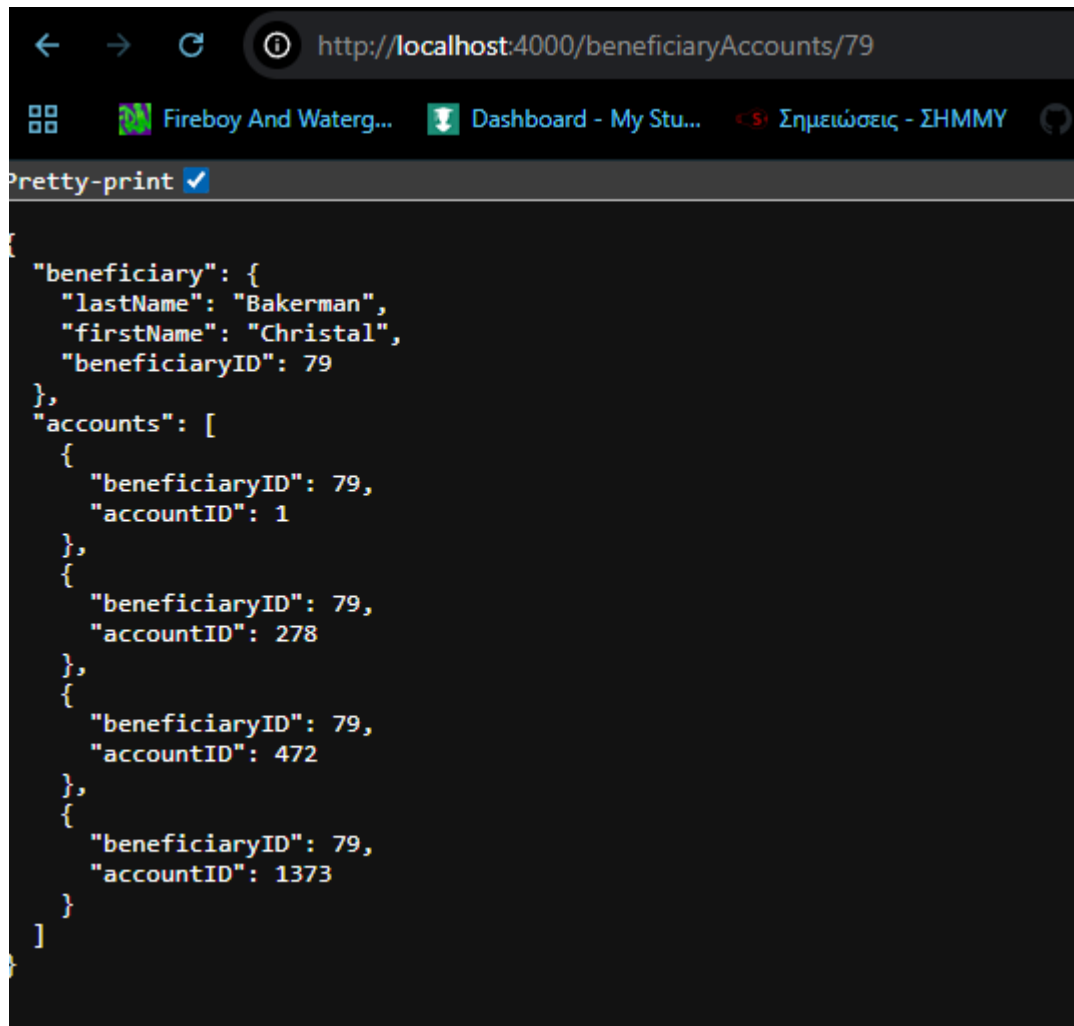
Exercise2.java

Το endpoint δέχεται ως **path parameter** το beneficiaryId.

Χρησιμοποιώντας τον αλγόριθμο **BinarySearchAccounts**, εντοπίζονται όλα τα accounts που σχετίζονται με τον συγκεκριμένο δικαιούχο.

Το αποτέλεσμα επιστρέφεται σε μορφή BeneficiaryWithAccounts object το οποίο περιλαμβάνει:

- τα στοιχεία του δικαιούχου (beneficiary),
- τη λίστα με όλους τους λογαριασμούς του (accounts).



```
{
  "beneficiary": {
    "lastName": "Bakerman",
    "firstName": "Christal",
    "beneficiaryID": 79
  },
  "accounts": [
    {
      "beneficiaryID": 79,
      "accountID": 1
    },
    {
      "beneficiaryID": 79,
      "accountID": 278
    },
    {
      "beneficiaryID": 79,
      "accountID": 472
    },
    {
      "beneficiaryID": 79,
      "accountID": 1373
    }
  ]
}
```

3. Ανάκτηση των συναλλαγών ενός δικαιούχου.

Exercise3.java

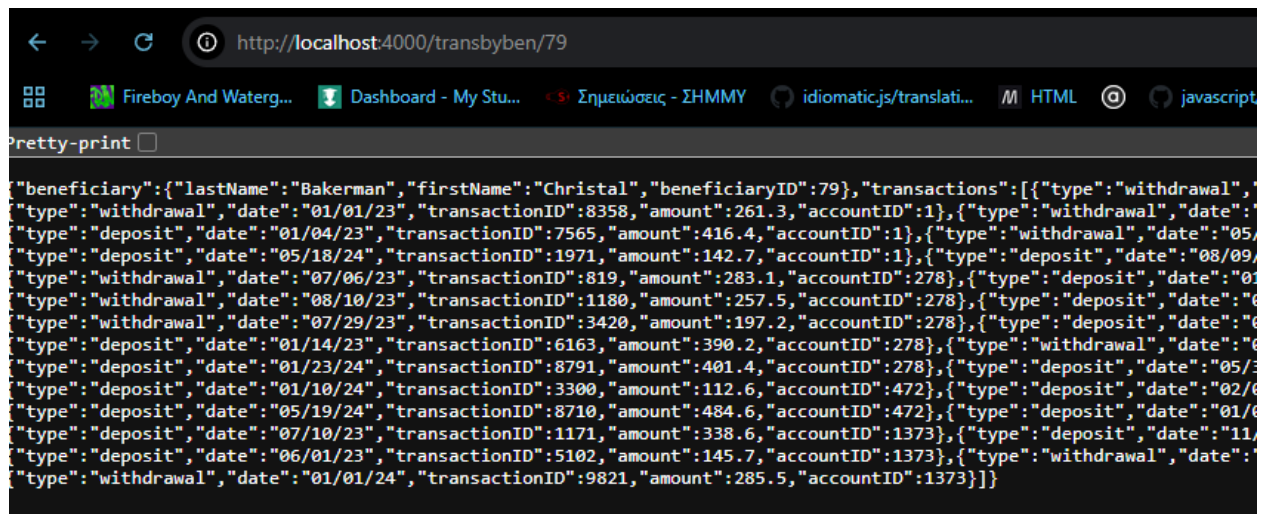
Το endpoint δέχεται ως **path parameter** το beneficiaryId.

Όπως και στην Άσκηση 2:

1. Εντοπίζουμε όλα τα accounts που συνδέονται με τον συγκεκριμένο δικαιούχο.
2. Συλλέγουμε τα accountIds σε μια λίστα.
3. Χρησιμοποιούμε τον αλγόριθμο **BinarySearchTransaction** ώστε να εντοπίσουμε αποδοτικά όλες τις συναλλαγές που σχετίζονται με αυτά τα accounts.

Το αποτέλεσμα επιστρέφεται σε μορφή BeneficiaryWithTransactions object, το οποίο περιλαμβάνει:

- τα στοιχεία του δικαιούχου (beneficiary),
- τη λίστα με όλες τις συναλλαγές του (transactions).



```
http://localhost:4000/transbyben/79

{"beneficiary":{"lastName":"Bakerman","firstName":"Christal","beneficiaryID":79},"transactions":[{"type":"withdrawal","date":"01/01/23","transactionID":8358,"amount":261.3,"accountID":1}, {"type":"withdrawal","date":"01/04/23","transactionID":7565,"amount":416.4,"accountID":1}, {"type":"deposit","date":"05/18/24","transactionID":1971,"amount":142.7,"accountID":1}, {"type":"deposit","date":"08/09/23","transactionID":819,"amount":283.1,"accountID":278}, {"type":"withdrawal","date":"07/06/23","transactionID":1180,"amount":257.5,"accountID":278}, {"type":"deposit","date":"07/29/23","transactionID":3420,"amount":197.2,"accountID":278}, {"type":"deposit","date":"01/14/23","transactionID":6163,"amount":390.2,"accountID":278}, {"type":"withdrawal","date":"01/23/24","transactionID":8791,"amount":401.4,"accountID":278}, {"type":"deposit","date":"05/10/24","transactionID":3300,"amount":112.6,"accountID":472}, {"type":"deposit","date":"02/05/24","transactionID":8710,"amount":484.6,"accountID":472}, {"type":"deposit","date":"01/07/23","transactionID":1171,"amount":338.6,"accountID":1373}, {"type":"deposit","date":"11/06/23","transactionID":5102,"amount":145.7,"accountID":1373}, {"type":"withdrawal","date":"01/01/24","transactionID":9821,"amount":285.5,"accountID":1373}]}
```

```
← → ↺ http://localhost:4000/transbyben/79
Fireboy And Waterg... Dashboard - My Stu... Σημειώσεις - ΣΗΜΜΥ idiomatic.js/translati... HTML @ j
Pretty-print ✓

{
  "beneficiary": {
    "lastName": "Bakerman",
    "firstName": "Christal",
    "beneficiaryID": 79
  },
  "transactions": [
    {
      "type": "withdrawal",
      "date": "03/13/24",
      "transactionID": 8943,
      "amount": 221.2,
      "accountID": 1
    },
    {
      "type": "withdrawal",
      "date": "01/01/23",
      "transactionID": 8358,
      "amount": 261.3,
      "accountID": 1
    },
    {
      "type": "withdrawal",
      "date": "06/07/24",
      "transactionID": 8066,
      "amount": 217.6,
      "accountID": 1
    },
    {
      "type": "deposit",
      "date": "01/19/23",
      "transactionID": 7791,
      "amount": 439.6,
      "accountID": 1
    },
    {
      "type": "deposit",
      "date": "01/04/23",
      "transactionID": 7565,
      "amount": 416.4,
      "accountID": 1
    },
    {
      "type": "withdrawal",
      "date": "05/15/24",
      "transactionID": 5902,
      "amount": 16.1,
      "accountID": 1
    },
    {
      "type": "deposit",
      "date": "10/12/23",
      "transactionID": 3632,
      "amount": 256.4,
      "accountID": 1
    },
  ]
}
```

4. Ανάκτηση του υπολοίπου των λογαριασμών ενός ανθρώπου.

Exercise4.java

Το endpoint δέχεται **POST request**.

Στο **body** της κλήσης περιμένει JSON με `firstName` και `lastName`.

Αναζητούμε στη λίστα με τους Beneficiaries τον δικαιούχο που αντιστοιχεί στο όνομα και επώνυμο.

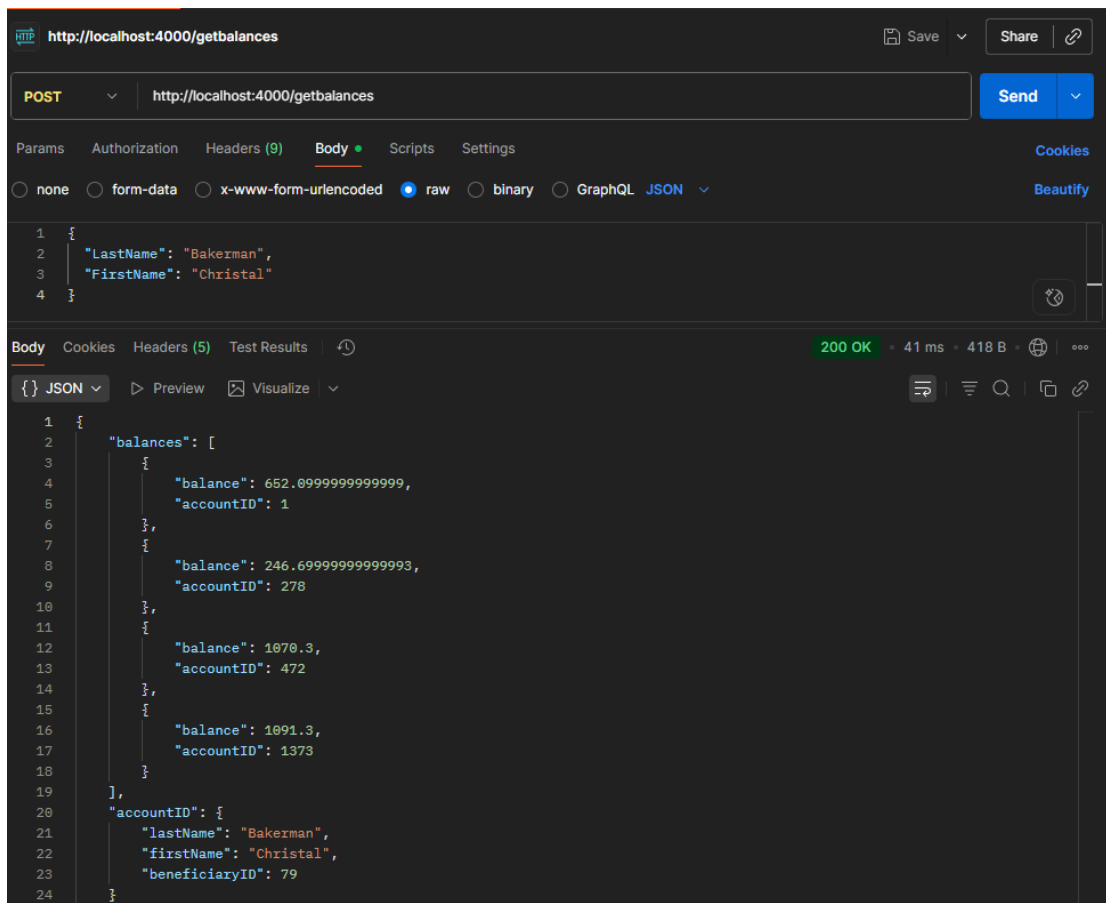
Παίρνουμε το `beneficiaryId` του συγκεκριμένου δικαιούχου.

Όπως και στην Άσκηση 3, βρίσκουμε όλα τα transactions που σχετίζονται με αυτόν, ταξινομημένα ως προς `accountId`.

Με τη βοήθεια του **BalanceCalculator**, υπολογίζουμε το υπόλοιπο κάθε λογαριασμού.

Το αποτέλεσμα επιστρέφεται σε μορφή `AccountWithBalances` object, το οποίο περιλαμβάνει:

- τα στοιχεία του δικαιούχου,
- λίστα από balances για κάθε λογαριασμό του.



5. Ανάκτηση της μεγαλύτερης ανάληψης για έναν δικαιούχο τον τελευταίο μήνα.

Exercise5.java

Το endpoint δέχεται ως **path parameter** το `beneficiaryId`.

Εντοπίζουμε όλα τα transactions που σχετίζονται με τον συγκεκριμένο δικαιούχο (όπως στις προηγούμενες ασκήσεις).

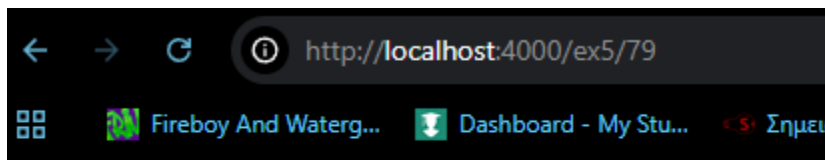
Φιλτράρουμε τη λίστα ώστε να μείνουν μόνο οι συναλλαγές τύπου **withdrawal**.

Από αυτές κρατάμε μόνο όσες έχουν ημερομηνία μέσα στον **τελευταίο μήνα** από την τρέχουσα ημερομηνία.

Επιστρέφουμε το transaction με το **μεγαλύτερο ποσό ανάληψης**.

Αν δεν υπάρχουν συναλλαγές που πληρούν τα παραπάνω κριτήρια, το endpoint επιστρέφει κατάλληλο μήνυμα.

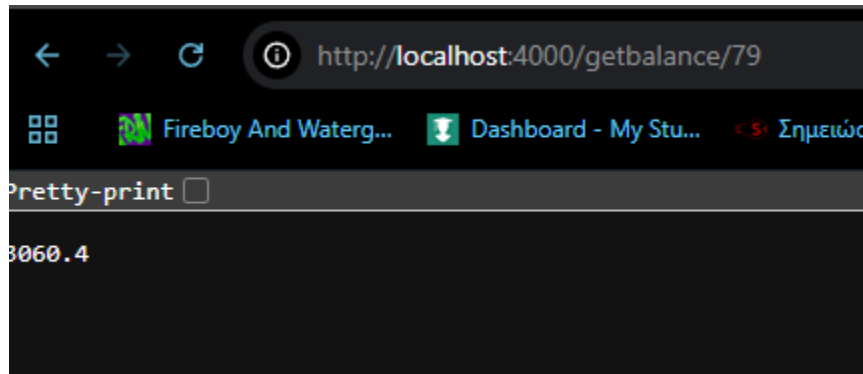
Επειδή τα transactions που φορτώνονται από το CSV αφορούν μόνο τα έτη **2023–2024**, ενώ η τρέχουσα ημερομηνία είναι **2025**, το endpoint θα επιστρέφει πάντα μήνυμα (“No withdrawals found...”) μέχρι να ανανεωθούν τα δεδομένα των transactions με πιο πρόσφατες ημερομηνίες.



No withdrawals found in the last month for beneficiary 79

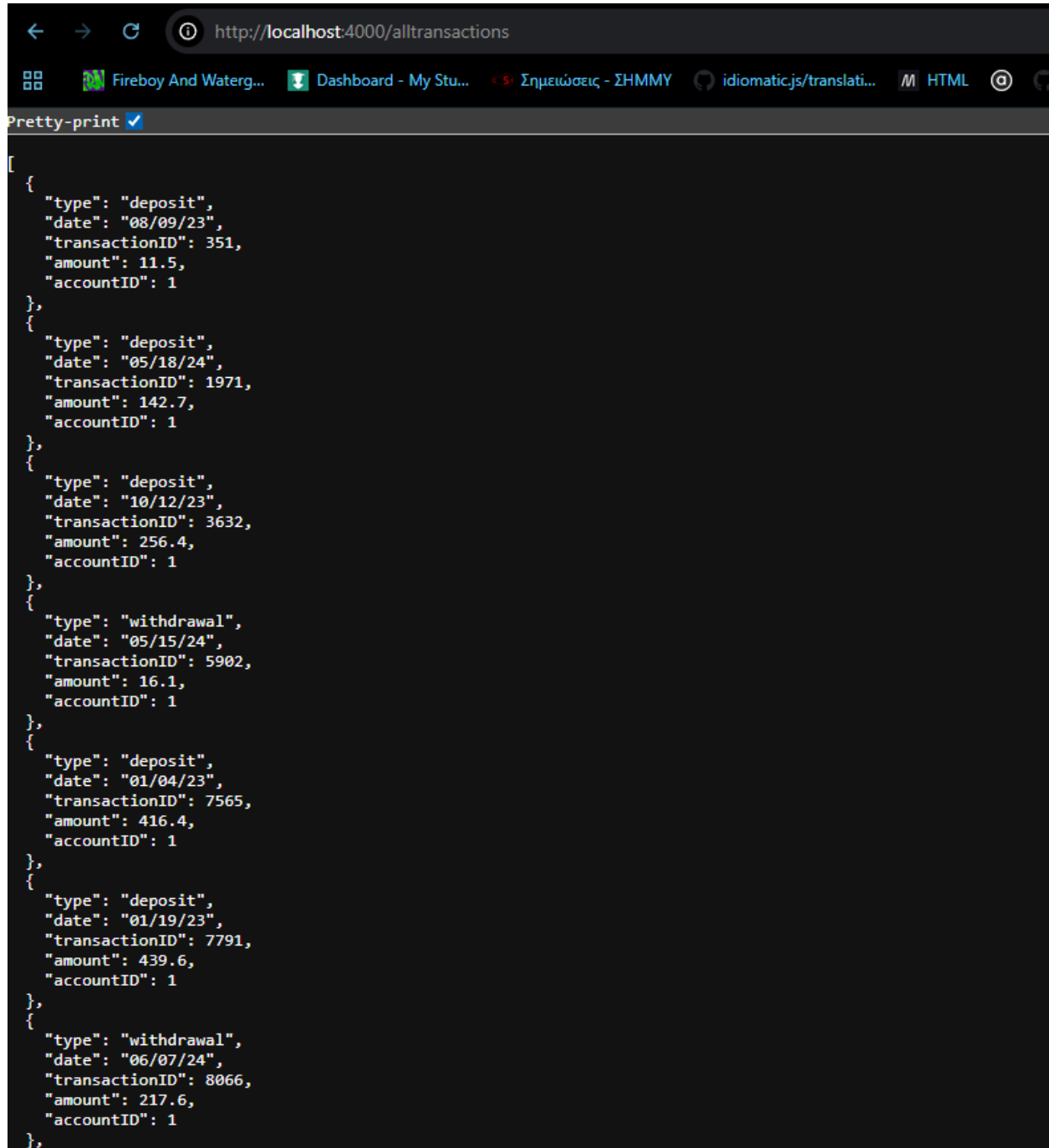
EXTRA ENDPOINTS ΠΟΥ ΕΙΧΑΝ ΦΤΙΑΧΤΕΙ :

Exercise4demo: επέστρεφε για έναν beneficiary το συνολικο balance που έχει αν σε όλους τους λογαριασμούς του προσθέταμε τα balances μαζί (δέχεται σαν path parameter το beneficiaryID)



AllTransactions:

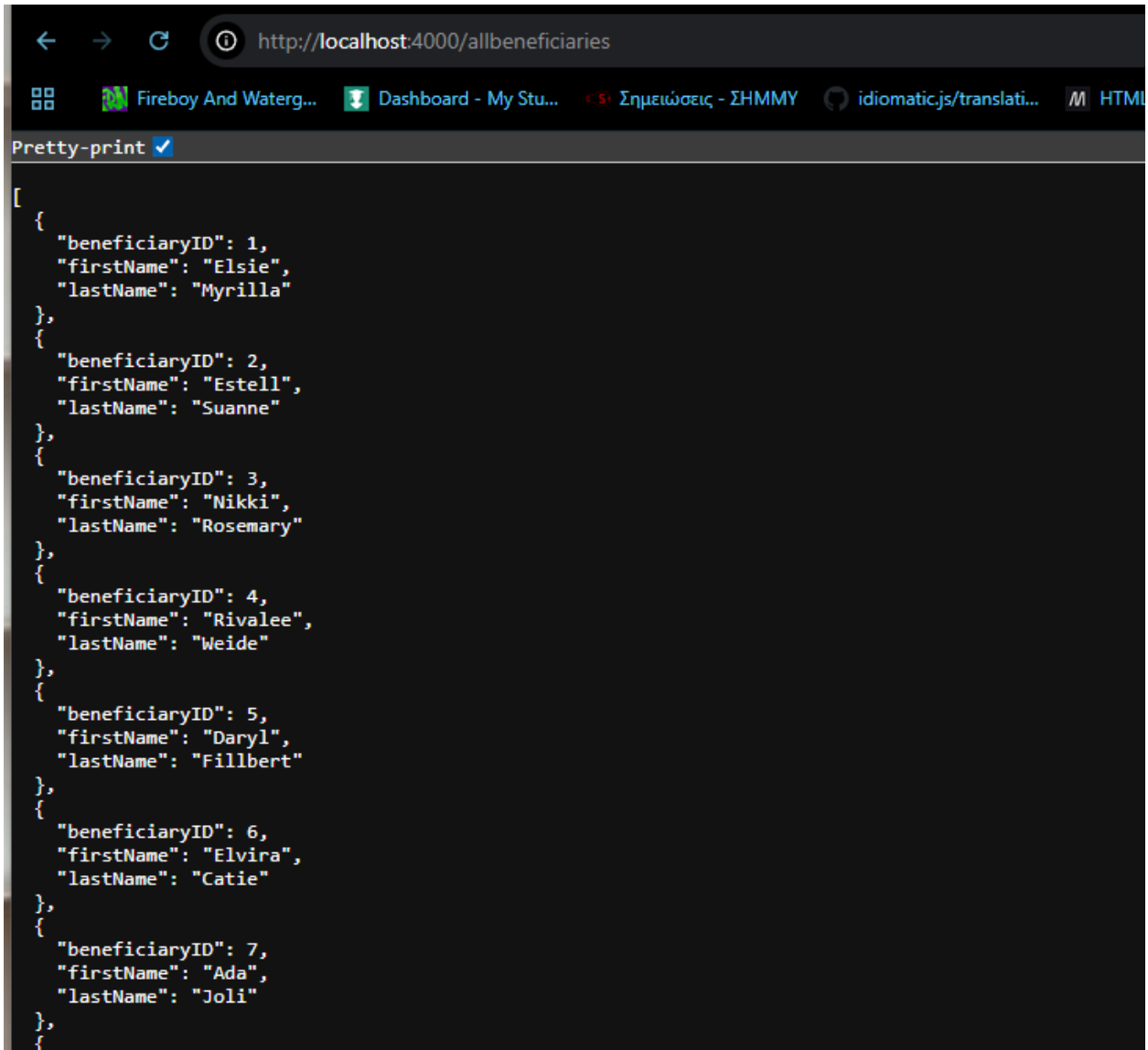
Επιστρέφει όλα τα transactions με αυξουσα σειρά κατά account id



The screenshot shows a web browser window with the address bar displaying `http://localhost:4000/alltransactions`. The browser's developer tools are open, showing a JSON array of transactions. The JSON is formatted with 'Pretty-print' checked. The array contains six objects, each representing a transaction with fields for type, date, transactionID, amount, and accountID. The transactions are sorted by accountID (all are 1) and then by transactionID in ascending order.

```
[
  {
    "type": "deposit",
    "date": "08/09/23",
    "transactionID": 351,
    "amount": 11.5,
    "accountID": 1
  },
  {
    "type": "deposit",
    "date": "05/18/24",
    "transactionID": 1971,
    "amount": 142.7,
    "accountID": 1
  },
  {
    "type": "deposit",
    "date": "10/12/23",
    "transactionID": 3632,
    "amount": 256.4,
    "accountID": 1
  },
  {
    "type": "withdrawal",
    "date": "05/15/24",
    "transactionID": 5902,
    "amount": 16.1,
    "accountID": 1
  },
  {
    "type": "deposit",
    "date": "01/04/23",
    "transactionID": 7565,
    "amount": 416.4,
    "accountID": 1
  },
  {
    "type": "deposit",
    "date": "01/19/23",
    "transactionID": 7791,
    "amount": 439.6,
    "accountID": 1
  },
  {
    "type": "withdrawal",
    "date": "06/07/24",
    "transactionID": 8066,
    "amount": 217.6,
    "accountID": 1
  },
]
```

AllBeneficiaries.java: επιστρέφει όλους του beneficiaries

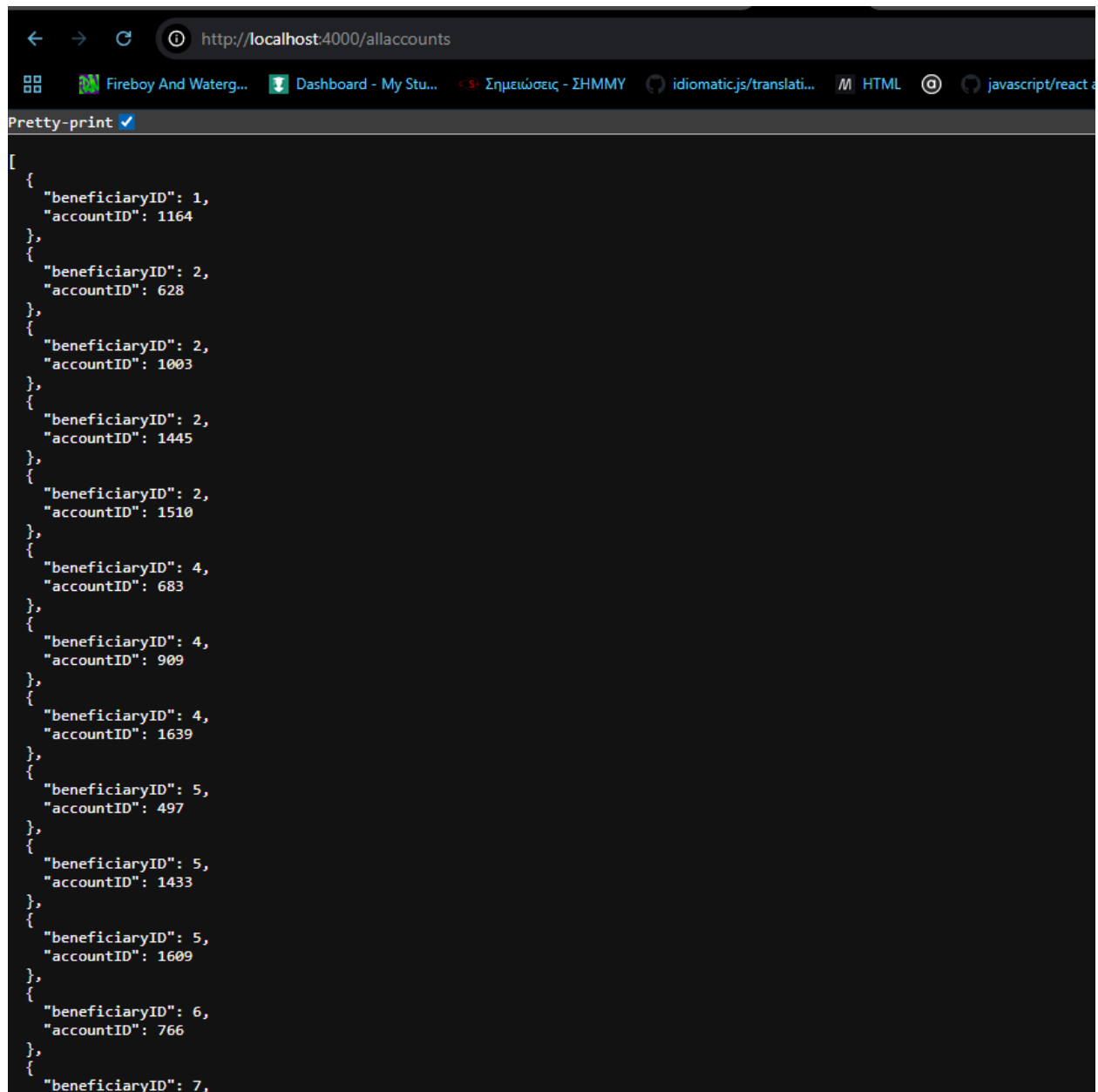


A screenshot of a web browser window displaying a JSON array of beneficiary data. The browser's address bar shows the URL `http://localhost:4000/allbeneficiaries`. The browser's tab bar includes several tabs: "Fireboy And Waterg...", "Dashboard - My Stu...", "Σημειώσεις - ΣΗΜΜΥ", "idiomatic.js/translati...", and "HTML". The browser's developer tools are open, showing the "Pretty-print" view of the JSON response. The JSON array contains seven objects, each representing a beneficiary with fields for "beneficiaryID", "firstName", and "lastName".

```
[
  {
    "beneficiaryID": 1,
    "firstName": "Elsie",
    "lastName": "Myrilla"
  },
  {
    "beneficiaryID": 2,
    "firstName": "Estell",
    "lastName": "Suanne"
  },
  {
    "beneficiaryID": 3,
    "firstName": "Nikki",
    "lastName": "Rosemary"
  },
  {
    "beneficiaryID": 4,
    "firstName": "Rivalee",
    "lastName": "Weide"
  },
  {
    "beneficiaryID": 5,
    "firstName": "Daryl",
    "lastName": "Fillbert"
  },
  {
    "beneficiaryID": 6,
    "firstName": "Elvira",
    "lastName": "Catie"
  },
  {
    "beneficiaryID": 7,
    "firstName": "Ada",
    "lastName": "Joli"
  },
  {

```

AllAccounts: επιστρέφει όλα τα accounts σε αυξουσα σειρα κατά Beneficiary ID



```
[
  {
    "beneficiaryID": 1,
    "accountID": 1164
  },
  {
    "beneficiaryID": 2,
    "accountID": 628
  },
  {
    "beneficiaryID": 2,
    "accountID": 1003
  },
  {
    "beneficiaryID": 2,
    "accountID": 1445
  },
  {
    "beneficiaryID": 2,
    "accountID": 1510
  },
  {
    "beneficiaryID": 4,
    "accountID": 683
  },
  {
    "beneficiaryID": 4,
    "accountID": 909
  },
  {
    "beneficiaryID": 4,
    "accountID": 1639
  },
  {
    "beneficiaryID": 5,
    "accountID": 497
  },
  {
    "beneficiaryID": 5,
    "accountID": 1433
  },
  {
    "beneficiaryID": 5,
    "accountID": 1609
  },
  {
    "beneficiaryID": 6,
    "accountID": 766
  },
  {
    "beneficiaryID": 7,
```