ARISTOTLE UNIVERSITY OF THESSALONIKI

Aristotle University of Thessaloniki
Department of Electrical and Computer Engineering

Αντάκης Μάριος 10619, *mantakis@ece.auth.gr*
Διακολουκάς Δημήτριος 10642, *ddiakolou@ece.auth.gr*
Καράτης Δημήτριος 10775, *karatisd@ece.auth.gr*

BankDB - Βάση Δεδομένων Τράπεζας

(ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ)

Τρίτο Παραδοτέο

Περιεχόμενα

1 Εισαγωγή

Το παρόν παραδοτέο αποτελεί το τελικό στάδιο της εργασίας. Σκοπός του είναι η ολοκλήρωση μιας δυναμικής web εφαρμογής η οποία διασυνδέεται με τη βάση δεδομένων BankDB που σχεδιάστηκε στα προηγούμενα στάδια. Η εφαρμογή παρέχει ένα ασφαλές και φιλικό περιβάλλον εργασίας για τους πελάτες της τράπεζας, επιτρέποντάς τους να διαχειρίζονται τους λογαριασμούς τους, να εκτελούν μεταφορές κεφαλαίων και να ενημερώνονται για τα προϊόντα τους (κάρτες, δάνεια) σε πραγματικό χρόνο.

2 Απαιτήσεις Συστήματος

Για την επιτυχή εκτέλεση της εφαρμογής απαιτούνται τα εξής προαπαιτούμενα:

- **Γλώσσα Προγραμματισμού:** Python 3.x.
- **Βιβλιοθήκες Python:**
  - `Flask`: Για την ανάπτυξη του web backend.
  - `mysql-connector-python`: Για τη διασύνδεση με τον MySQL Server.
- **Σύστημα Διαχείρισης Βάσεων Δεδομένων:** MySQL Server (τοπικά ή στο Cloud).

3 Οδηγίες Εγκατάστασης και Εκτέλεσης

Η εφαρμογή ακολουθεί τη δομή του Flask framework. Τα βήματα για την εκτέλεση είναι:

1. **Προετοιμασία Βάσης:** Δημιουργία της βάσης `BankDB` και εισαγωγή του παρεχόμενου SQL dump αρχείου (`Dump20251219.sql`).
2. **Ρύθμιση Σύνδεσης:** Παραμετροποίηση του `db_config` στο αρχείο `app.py` με τα κατάλληλα credentials (Host, User, Password).
3. **Εκκίνηση Εφαρμογής:** Εκτέλεση της εντολής `python3 app.py` από τον φάκελο του project.
4. **Πρόσβαση:** Η εφαρμογή είναι προσβάσιμη μέσω web browser στη διεύθυνση `http://127.0.0.1:5000`.

Πλήρως αναλυτικός οδηγός εκτέλεσης της εφαρμογής καθώς και παραπάνω πληροφορίες βρίσκονται στο `README.txt`.

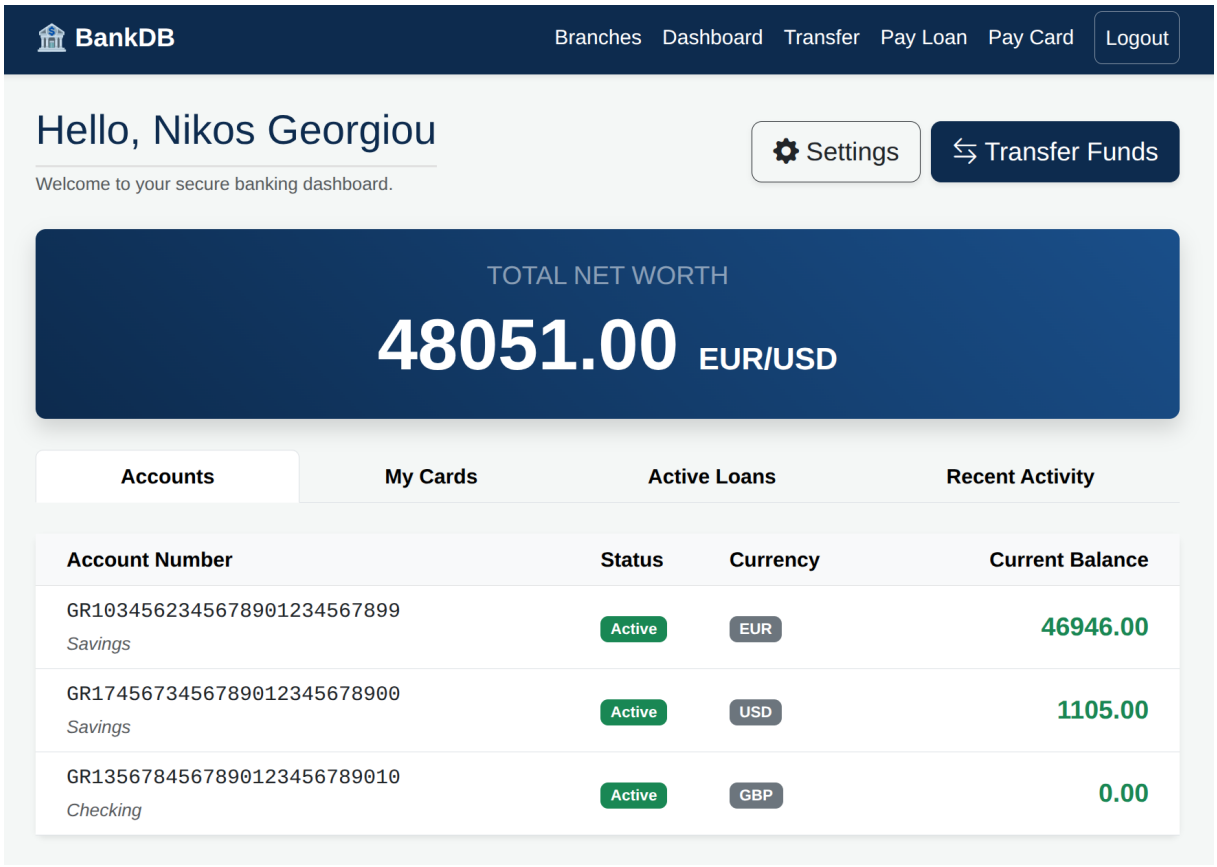
4 Εργαλεία και Τεχνολογίες

- **Backend (Python Flask):** Επιλέχθηκε για την ελαφριά του δομή και την ευκολία στη διαχείριση.
- **Frontend (Bootstrap 5 & Jinja2):** Για τη διεπαφή του χρήστη επιλέχθηκε ο συνδυασμός του Bootstrap 5 και της μηχανής προτύπων (templating engine) Jinja2.
  - Το **Bootstrap 5** χρησιμοποιήθηκε για να διασφαλιστεί μια επαγγελματική και responsive εμφάνιση, επιτρέποντας στους πίνακες των λογαριασμών και στις φόρμες των συναλλαγών να προσαρμόζονται αυτόματα σε κάθε μέγεθος οθόνης.
  - Χρησιμοποιήθηκε η **Jinja2**, η οποία αποτελεί την προεπιλεγμένη (default) μηχανή προτύπων του Flask. Επιτρέπει τη δυναμική παραγωγή του HTML κώδικα διοχετεύοντας δεδομένα απευθείας από τη βάση (π.χ. όνομα πελάτη, λίστα συναλλαγών) στα πρότυπα κατά τον χρόνο εκτέλεσης. Με αυτόν τον τρόπο διαχωρίζεται πλήρως η λογική του backend από το frontend, αποφεύγοντας τη χρήση στατικών σελίδων.
- **MySQL Views:** Η εφαρμογή βασίζεται σε SQL Views (`accounts_balance`, `customer_accounts`, `loan_debts`, κ.α.) για τον υπολογισμό των υπολοίπων κλπ, μεταφέροντας το υπολογιστικό βάρος από τον κώδικα Python απευθείας στη βάση δεδομένων.

5 Ανάλυση Λειτουργικότητας και Κώδικα

5.1 Βασικές Λειτουργίες Πελάτη

- **Login:** Αυθεντικοποίηση μέσω του ΑΦΜ (TIN). Δεν απαιτείται σταθερός κωδικός πρόσβασης για την επίδειξη.
- **Dashboard:** Ενοποιημένη προβολή όλων των λογαριασμών (Ενεργών, Κλειστών κλπ), καρτών και δανείων.
- **Διαχείριση Προφίλ (Settings):** Δυνατότητα αλλαγής διεύθυνσης κατοικίας, πλήρης διαχείριση πολλαπλών emails (προσθήκη, επεξεργασία, διαγραφή) καθώς και εμφάνιση ονόματος, τηλεφώνων και ΑΦΜ (χωρίς τη δυνατότητα επεξεργασία τους μέσω της εφαρμογής, αλλά μόνο απευθείας μέσω καταστήματος).
- **Branch Locator:** Δυναμική λίστα καταστημάτων με τα στοιχεία επικοινωνίας τους.



Σχήμα 1: Dashboard BankDB web-app πελάτη με ΑΦΜ: 987654321

5.2 Λογική Συναλλαγών

Η εφαρμογή δίνει ιδιαίτερη έμφαση στην ακεραιότητα των δεδομένων χρησιμοποιώντας **ACID Transactions**. Η προσέγγιση αυτή διασφαλίζει ότι κάθε τραπεζική συναλλαγή (όπως η μεταφορά κεφαλαίων) εκτελείται με βάση τις εξής τέσσερις αρχές:

- **Atomicity (Ατομικότητα):** Η συναλλαγή αντιμετωπίζεται ως μία αδιαίρετη μονάδα. Είτε θα ολοκληρωθούν επιτυχώς όλα τα βήματα (χρέωση πηγής και πίστωση προορισμού), είτε δεν θα εκτελεστεί κανένα. Αν προκύψει σφάλμα στο ενδιάμεσο, το σύστημα εκτελεί `rollback`, επαναφέροντας τα υπόλοιπα στην αρχική τους κατάσταση.
- **Consistency (Συνέπεια):** Διασφαλίζεται ότι η βάση δεδομένων μεταβαίνει από μια έγκυρη κατάσταση σε μια άλλη, τηρώντας όλους τους περιορισμούς (π.χ. το υπόλοιπο δεν μπορεί να γίνει αρνητικό αν δεν επιτρέπεται).
- **Isolation (Απομόνωση):** Κάθε συναλλαγή εκτελείται ανεξάρτητα. Αν δύο χρήστες μεταφέρουν χρήματα ταυτόχρονα, το σύστημα διαχειρίζεται τις εγγραφές σειριακά, αποτρέποντας τη σύγχυση στα υπόλοιπα των λογαριασμών.
- **Durability (Ανθεκτικότητα):** Μόλις η συναλλαγή ολοκληρωθεί επιτυχώς (`commit`), τα δεδομένα εγγράφονται μόνιμα στη βάση και δεν

χάνονται ακόμη και σε περίπτωση διακοπής ρεύματος ή κατάρρευσης του server.

5.2.1 Μεταφορά Κεφαλαίων

Χρησιμοποιείται η μέθοδος `conn.start_transaction()` για να διασφαλιστεί ότι η αφαίρεση χρημάτων από τον λογαριασμό πηγής και η πίστωση στον λογαριασμό προορισμού θα γίνουν ως μία ενιαία ενέργεια. Σε περίπτωση οποιουδήποτε σφάλματος, εκτελείται `rollback`.

5.2.2 Πληρωμή Πιστωτικής Κάρτας (Double Transaction Strategy)

Η πιο σύνθετη λειτουργία του κώδικα αφορά την πληρωμή πιστωτικής κάρτας (`pay_credit`). Λόγω της δομής των SQL Views που υπολογίζουν το υπόλοιπο, εφαρμόστηκε μια στρατηγική διπλής εγγραφής:

1. **Εγγραφή T1 (Θετική):** Εισάγεται μια συναλλαγή με θετικό ποσό (`+amount`) που συνδέεται με την κάρτα. Αυτό αυξάνει το διαθέσιμο υπόλοιπό της.
2. **Εγγραφή T2 (Αρνητική):** Εισάγεται μια συναλλαγή με το διπλάσιο αρνητικό ποσό (`-2 * amount`) που συνδέεται μόνο με τον τραπεζικό λογαριασμό.

Αποτέλεσμα: Ο λογαριασμός χρεώνεται τελικά με το σωστό ποσό (`+amount - 2*amount = -amount`), ενώ η κάρτα πιστώνεται κανονικά. Για την αποφυγή σύγχυσης στο Dashboard, ο κώδικας διαιρεί οπτικά το ποσό διά 2 στην προβολή του ιστορικού (`CC_Repayment`).

5.2.3 Πληρωμή Δανείου

Για την αποπληρωμή δόσης δανείου, η εφαρμογή ακολουθεί μια διαδικασία ελέγχου και ενημέρωσης δύο επιπέδων. Αρχικά, το σύστημα επαληθεύει ότι ο επιλεγμένος λογαριασμός διαθέτει επαρκές υπόλοιπο. Στη συνέχεια, μέσω ενός ACID transaction, ο κώδικας εισάγει μια νέα εγγραφή στον πίνακα `transaction` με αρνητικό πρόσημο, η οποία συνδέεται με το συγκεκριμένο `LoanID`.

Η δυναμική ενημέρωση του οφειλόμενου ποσού επιτυγχάνεται μέσω του SQL View `loan_debts`. Το συγκεκριμένο view επανυπολογίζει το τρέχον χρέος (Debt) εκτελώντας το άθροισμα των συναλλαγών σε πραγματικό χρόνο. Έτσι, διασφαλίζεται ότι οποιαδήποτε πληρωμή καταχωρείται στη βάση αντανακλάται άμεσα στην εικόνα του πελάτη χωρίς την ανάγκη χειροκίνητης ενημέρωσης επιπλέον πινάκων.

6 Ασφάλεια και Αξιοπιστία Εφαρμογής

- **SQL Injection Protection:** Όλα τα queries εκτελούνται με `Prepared Statements` (χρήση παραμέτρων %s), αποτρέποντας κακόβουλες παρεμβάσεις.
- **Manual Auto-Increment:** Επειδή το σχήμα της βάσης δεν προέβλεπε αυτόματη αύξηση στο `TransactionID`, η εφαρμογή υπολογίζει δυναμικά το επόμενο ID μέσω της συνάρτησης `MAX(TransactionID) + 1`, διασφαλίζοντας τη μοναδικότητα των εγγραφών.
- **Session Management:** Χρήση του `session` της Flask για τον έλεγχο πρόσβασης μόνο σε εξουσιοδοτημένους χρήστες.

7 Διαθέσιμοι Χρήστες για Δοκιμή Συστήματος

Για την ευκολία ελέγχου και την επίδειξη των λειτουργιών της εφαρμογής, παρατίθεται ο πλήρης πίνακας των εγγεγραμμένων πελατών και των λογαριασμών τους, όπως αυτοί ανακτώνται δυναμικά από το view `customer_accounts`.

CustomerID	**Ονοματεπώνυμο**	**TIN (Username)**	**Αριθμός Λογαριασμού**
1	Maria Papadopoulou	123456789	GR1634441234567890123456789
1	Maria Papadopoulou	123456789	GR1311891434567890123456789
2	Maria Papadopoulou	321456789	GR1223459876543210123456789
3	Nikos Georgiou	987654321	GR1034562345678901234567899
3	Nikos Georgiou	987654321	GR1745673456789012345678900
3	Nikos Georgiou	987654321	GR1356784567890123456789010
4	Elena Dimitriou	112233445	GR1167895678901234567890121
5	Kostas Vassilakis	223344556	GR1478906789012345678901231
5	Kostas Vassilakis	223344556	GR1589017890123456789012342
6	Yiannis Papadakis	334455667	GR1990128901234567890123453
6	Yiannis Papadakis	334455667	GR2001239012345678901234564

Πίνακας 1: Συνολική λίστα πελατών και συνδεδεμένων λογαριασμών.

- **Συγχρονισμός Δεδομένων:** Επισημαίνεται ότι οποιαδήποτε αλλαγή μέσα από το web-app (π.χ. μεταφορά ποσού από έναν λογαριασμό σε έναν άλλο) θα ενημερώσει άμεσα τα αντίστοιχα πεδία στη βάση δεδομένων και αντίστροφα.

8 Live Web Deployment & Cloud Hosting

Για λόγους πληρότητας η εφαρμογή φιλοξενείται ζωντανά και στο διαδίκτυο μέσω των παρακάτω υπηρεσιών Cloud:

- **Render:** Χρησιμοποιήθηκε για τη φιλοξενία του Web Server (Flask backend).
- **Aiven Console:** Χρησιμοποιήθηκε για τη διαχείριση της MySQL βάσης δεδομένων σε Cloud περιβάλλον.

8.1 Προσαρμογές για το Cloud Περιβάλλον

Προκειμένου να επιτευχθεί η επιτυχής σύνδεση και εκτέλεση στο Cloud, πραγματοποιήθηκαν οι εξής κρίσιμες αλλαγές:

1. **Κώδικας Python (`app.py`):** Ενσωματώθηκε υποστήριξη για κρυπτογραφημένη σύνδεση SSL (`ssl_disabled=Φαλσε`) και χρήση μεταβλητών περιβάλλοντος (`Environment Variables`) αντί για hardcoded credentials. Επίσης, ρυθμίστηκε δυναμικό `PORT` για συμβατότητα με τον διακομιστή του Render.
2. **SQL Dump:** Το αρχικό dump τροποποιήθηκε ώστε να αφαιρεθούν οι εντολές `DEFINER`, οι οποίες περιόριζαν τη δημιουργία των Views αποκλειστικά στον τοπικό χρήστη `root`.

8.2 Αξιολόγηση Απόδοσης

Παρά την επιτυχή λειτουργία στο διαδίκτυο, παρατηρείται ότι η ταχύτητα απόκρισης είναι αισθητά χαμηλότερη σε σχέση με την τοπική εκτέλεση, λόγω της καθυστέρησης (latency) στη μεταφορά δεδομένων μεταξύ των Cloud servers. Ως εκ τούτου, **η βέλτιστη εμπειρία χρήσης και αξιολόγησης παραμένει η τοπική εκτέλεση μέσω Localhost**.

8.3 Συμπεράσματα Προσέγγισης

Αυτός ο τρόπος προσέγγισης (Cloud Hybrid Model) επιτρέπει την πλήρη αποσύνδεση του Backend από το Database Server, προσφέροντας υψηλή διαθεσιμότητα και δυνατότητα πρόσβασης από οποιαδήποτε συσκευή. Ωστόσο, εισάγει προκλήσεις ασφαλείας που αντιμετωπίστηκαν εν μέρει με τη χρήση SSL και Prepared Statements.

8.4 Σύνδεσμος Εφαρμογής

Η Live Web εφαρμογή είναι διαθέσιμη για δοκιμή στην παρακάτω διεύθυνση: https://flask-mysql-bank-project.onrender.com/

- **Σημείωση Καθυστέρησης:** Λόγω της χρήσης του δωρεάν πακέτου φιλοξενίας (Render Free Tier), η εφαρμογή τίθεται σε κατάσταση αναμονής (sleep mode) μετά από περιόδους αδράνειας. Κατά την πρώτη επίσκεψη, ενδέχεται να υπάρξει μια καθυστέρηση της τάξης των **30-50 δευτερολέπτων** μέχρι να εκκινηθεί εκ νέου ο διακομιστής.
- **Διαθεσιμότητα Βάσης Δεδομένων:** Η βάση δεδομένων φιλοξενείται στην Aiven Console. Σε περίπτωση παρατεταμένης αδράνειας, η υπηρεσία ενδέχεται να θέσει τη βάση σε κατάσταση Powered Off. Σε τέτοια περίπτωση, η εφαρμογή θα εμφανίσει σφάλμα σύνδεσης μέχρι τη χειροκίνητη επανεκκίνηση (Power On) της υπηρεσίας από τον διαχειριστή.

Αναφορές

[1] Υλικό και Διαλέξεις Μαθήματος Βάσεις Δεδομένων, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (ΑΠΘ), Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών.

[2] Flask Documentation, Official User Guide for Web Development with Python, https://flask.palletsprojects.com/.

[3] MySQL Connector/Python Developer Guide, Oracle Corporation, https://dev.mysql.com/doc/connector-python/en/.

[4] Bootstrap v5.3 Documentation, Responsive Front-End Framework, https://getbootstrap.com/docs/5.3/.

[5] Jinja Templating Engine for Python, Official Documentation, https://jinja.palletsprojects.com/.

[6] Render Documentation, Cloud Application Hosting Platform, https://render.com/docs.

[7] Aiven Console Documentation, Managed MySQL Database Services, https://docs.aiven.io/.