[Skills Test] Back-end interns & apprentices 2023

Introduction

Hi! First things first: thanks a lot for taking the time to do this coding exercise.

We know your time is valuable and we really appreciate your commitment. We hope you will feel comfortable writing this test and submit work you will be proud of.

It typically takes **1 hour to complete**. If you need more time or less time, please do not hesitate to mention why.

You may use the language of your choice between Go, Ruby, Python, Java, and JavaScript. Ask us if you'd like to do it in another language.

On behalf of the whole Backend Team at Qonto, we thank you once again, and wish you to succeed in this test!

Functional exercises

💡**TODO** : For each question, please write expected outcomes or follow ups.

```
# Write a function that determines if any of its arguments is true.
[true, false] => true
[1, 0, "string"] => false

# Write a function that returns a list of even integers from 0 to n inclusive.
n = 7 => [0, 2, 4, 6]

# Write a function that counts how many times each item occurs in a list.
["egg", "egg", "soap", "soap", "soap"] => {"egg" => 2, "soap" => 3}

# Write a function that returns true if a student went to that college
alison = { college: :cambridge, year: 2008 } :cambridge => true
brian  = { college: :oxford, year: 2005 } :cambridge => false
```

```
chris  = { college: :cambridge, year: 2002 } :oxford => false
danni  = { college: :cambridge, year: 2004 } :cambridge => true

# Update that function to return true if a student attended college during set start and end date
alison = { college: :cambridge, year: 2008 } => false
brian  = { college: :oxford, year: 2005 } => false
chris  = { college: :cambridge, year: 2002 } => false
danni  = { college: :cambridge, year: 2004 } => true

# Write a function that returns the integer value of a string argument
"2" => 2
"10.0" => 10
"hello 15" => 15
```

💡 *How to read code*

▼ What does this function do?

```
func (dbClient *Store) DoSomething(
  ctx context.Context,
  tx *dbHelper.DB,
  iban string,
  startDate time.Time
) (db.Account, int, error) {
  if !validator.IsIBAN(&iban) {
    return db.Account{}, 0, errors.New("invalid iban")
  }

  response := struct {
    TxsUpdated int
  }{}

  if err := tx.Exec(`
      SET LOCAL lock_timeout TO 0; -- disabled
      SET LOCAL statement_timeout TO 0; -- disabled
    `).Error; err != nil {
    return db.Account{}, 0, errors.Annotate(err, "failed to disable timeouts")
  }

  if err := tx.Raw(RecalculateTransactionBalances, iban, startDate).Scan(&response).Error;
err != nil {
    return db.Account{}, 0, errors.Annotate(err, "failed to recalculate balances")
  }

  if err := tx.Exec("SELECT * FROM accounts WHERE iban = $1 FOR UPDATE;", iban).Error; err
!= nil {
    return db.Account{}, 0, errors.Annotate(err, "failed to select account")
  }

  lastBalance := struct {
    Balance int64
  }{}
  err := tx.Raw(LastAccountedBalance, iban).Scan(&lastBalance).Error // lastBalance = get_
last_accounted_balance

  if dbutil.IsRecordNotFoundError(err) {
    account, err := dbClient.GetAccount(ctx, "", iban, "")
    if err != nil {
      return db.Account{}, 0, errors.New("failed to get an account")
    }
    return *account, response.TxsUpdated, nil
  }

  if err != nil {
    return db.Account{}, 0, errors.Annotate(err, "could not get last accounted balance")
  }

  updateBalance := `
      UPDATE accounts
      SET
        balance = $1::BIGINT,
        updated_at = NOW()
      WHERE iban = $2
      RETURNING *
    `

  account := db.Account{}
  err = tx.Raw(updateBalance, lastBalance.Balance, iban).Scan(&account).Error

  if err != nil {
    return db.Account{}, 0, errors.Annotate(err, "could not update account balance")
  }

  return account, response.TxsUpdated, nil
}
```