//
Av. Leandro N. Alem 619
Piso #8
Ciudad de Buenos Aires
Argentina

//
T +54 11 5352-1259
info@atixlabs.com
//
www.atixlabs.com

ATIX LABS _

TECHNICAL CHALLENGE - Backend Dev

> REMOTE

INTRODUCTION _

The following challenge goal is to assess your skills as a developer.

The following items will be taken in consideration:

- Delivering something it's a good start.
- We don't expect from you to know all the answers beforehand. Researching is a great skill.
- If you cannot solve an item, don't worry, we can always learn new things later. It's a plus if you share with us any partial result or idea in order to understand your thinking process.
- It's important not to miss the deadline.

//ATIX LABS

//
Av. Leandro N. Alem 619
Piso #8
Ciudad de Buenos Aires
Argentina

//
T +54 11 5352-1259
info@atixlabs.com
//
www.atixlabs.com

1. CODING _

A. Design

Given the following code, which represents an Argentinian Bank account, please answer the following question:

i. Do you think the class is properly designed? If not, but it works, would you change it or will you add new classes?
ii. If you would change the design, please explain what you would change using an UML class diagram. If you think a method should be changed, please write it down as well.
iii. Are these changes riskless? Why? How would you introduce them?
iv. Would you add any getters and setters? Which ones? Why?
v. Does your solution apply any design pattern? Which one?

```
public class Cuenta {
    public static final int CTA_CORRIENTE = 0;
    public static final int CAJA_AHORRO = 1;
    private int tipo;
    private long numeroCuenta;
    private String titular;
    private long saldo;
    private long descubiertoAcordado;

    public Cuenta(int tipo, long nCuenta, String titular, long descAcordado) {
        this.tipo = tipo;
        this.numeroCuenta = nCuenta;
        this.titular = titular;
        if (tipo == CTA_CORRIENTE)
            this.descubiertoAcordado = descAcordado;
        else this.descubiertoAcordado = 0;
        saldo = 0;
    }
    public Cuenta(int tipo, long numeroCuenta, String titular) {
        this.tipo = tipo;
        this.numeroCuenta = numeroCuenta;
        this.titular = titular;
        this.descubiertoAcordado = 0;
        saldo = 0;
    }
    public void depositar(long monto) {
```

//
Av. Leandro N. Alem 619
Piso #8
Ciudad de Buenos Aires
Argentina

//
T +54 11 5352-1259
info@atixlabs.com
//
www.atixlabs.com

```
        saldo += monto;
    }

    public void extraer(long monto) throws RuntimeException {
        switch (tipo) {
            case CAJA_AHORRO:
                if (monto > saldo)
                    throw new RuntimeException("No hay dinero suficiente");
            case CTA_CORRIENTE:
                if (monto > saldo + descubiertoAcordado)
                    throw new RuntimeException("No hay dinero suficiente");
        }
        saldo -= monto;
    }
}
```

B. Programming: Linked line common log service

GOAL _

It's required to program a web service to allow users to, when invoking an API method (POST), to write on a shared log file such that each entry (line) is linked to the previous one using it's hash and a proof of work.

Expected line output CSV (file on server):

`` `prev_hash,message,nonce` ``

where:

- ***prev_hash*:** previous line hash (sha256) in hex format without any separators. You should use random for the first line.
- ***message*:** message sent by the user.
- ***nonce*:** a number that guarantees that `` `sha256(pre_hash + message + nonce) => RegEx('^00.*')` ``, i.e., starts with two zeroes.

//ATIX LABS

Av. Leandro N. Alem 619
Piso #8
Ciudad de Buenos Aires
Argentina

T +54 11 5352-1259
info@atixlabs.com
www.atixlabs.com

Log file content example:

```
0000000000000000000000000000000000000000000000000000000000000000,Hola Mundo,5
0038711c83bd05b1e369e27246df4ba815a6dda04116b1b2f9a8c21ba4e1de38,Chau Mundo,71
```

Where effectively it holds that:

```
sha256(pre_hash + message + nonce) => RegEx('^00.*')
```

For example, you can find in the second line:

```
0038711c83bd05b1e369e27246df4ba815a6dda04116b1b2f9a8c21ba4e1de38,Chau Mundo,71
```

If we want to verify the nonce is valid, we need to:

```
>   HEX(SHA256(0038711c83bd05b1e369e27246df4ba815a6dda04116b1b2f9a8c21ba4e1de38,Chau Mundo,71))
> 00232c7d3c2283695a4029eddc1b9e8c83914515832a04f57b402fc444aa11b5
```

In NodeJs you could calculate it doing:

```
crypto
  .createHash('sha256')
    .update('0038711c83bd05b1e369e27246df4ba815a6dda04116b1b2f9a8c21ba4e1de38,Chau Mundo,71', 'utf8')
  .digest('hex')
  .toString()
```

SUCCESS CRITERIA _

- Code needs to be published on Github or Gitlab. We will be reviewing commit history.
- It must contain a proper README file to help reviewers to understand how to launch or test the application. Also, don't forget to keep your code clean and easy to read.
- Users can perform requests (API calls) at the same time, the file log not necessarily must be sorted by requests income time.
- The file:

//
Av. Leandro N. Alem 619
Piso #8
Ciudad de Buenos Aires
Argentina

//
T +54 11 5352-1259
info@atixlabs.com
//
www.atixlabs.com

- The link between lines must hold for all the lines in the file, i.e., `***hash(line(n-1)) == line(n)[0])***`
- All the log lines hash must start with two zeroes.

NICE TO HAVE _

- Integration tests.
- Proper app logs to be able to troubleshoot.
- A single request must not block the file. All the requests must compete with each other to write down in the log file.

CONTEXT & FURTHER NOTES_

- Proof of Work: it's a cryptographic puzzle that, based on the dispersion (randomness) of hash functions, can be used to easily check a certain amount of work has been made as there is no other way to solve it besides doing bruteforce look up. It initially was created as a spam filter mechanism and nowadays is heavily used in blockchain applications.
- A consequence of hash-linking entries is that if anyone wants to edit one, all the hashes after it might be recalculated, i.e, all the work already made needs to be done again therefore it's harder (in terms of computing power) to change old entries. This is a way to encourage immutability.

2. QUESTIONS _

a. Explain the use of Strategy Pattern. How many instances are needed for each strategy? Is there any other pattern that might help? If so, please write down a simple example to demonstrate.
b. Please enumerate all the benefits of writing automated unit tests before writing the code.
c. When can Observer Pattern be used? What are the advantages of doing so?

3. SQL and DBS

Given the following tables (linked by idUsuario field):

//ATIX LABS

//
Av. Leandro N. Alem 619
Piso #8
Ciudad de Buenos Aires
Argentina

//
T +54 11 5352-1259
info@atixlabs.com
//
www.atixlabs.com

Usuario

```
id       integer
username varchar(255)
password varchar(255)
```

Persona

```
id        integer
idUsuario integer
nombre    varchar(255)
apellido  varchar(255)
fechaNac  date
```

Write a standard SQL query such that:

a. Returns the users whose name starts with “Jorg”
b. Return the months of the year where the amount of users who were born is bigger than 10.

* you can assume that there is a function called MONTH that returns the month of the year for a given date.