Avalanche Overnight

Trillions Moved, Minimal Costs

Powered by

Our Team

Gustavo Sanchez

Full Stack Developer
Business Specialist
Cumputer Engineering @ USP

Pedro Peres

Blockchain Developer
Full Stack Developer
Aeronautical Engineering @ USP

Ryan Viana

Blockchain Developer
Full Stack Developer
Cumputer Engineering @ USP

Reserve Requirements and Overnight Operation

Objectives

- Preserve excessive leverage;
- Control of Monetary Liquidity;

Collateral

- Interbank credit operations;
- T-Bills and T-Bonds;
- Repo Market

Around the World

- United States - FED;
- Europe - ECB;
- Brazil - BCB;

Overnight in Brazil

One of the most relevant activities inside Brazil's financial market

1

Central Bank of Brazil mandatory operation

Overnight operations are the linchpin of financial stability, balancing liquidity and regulatory compliance daily.

2

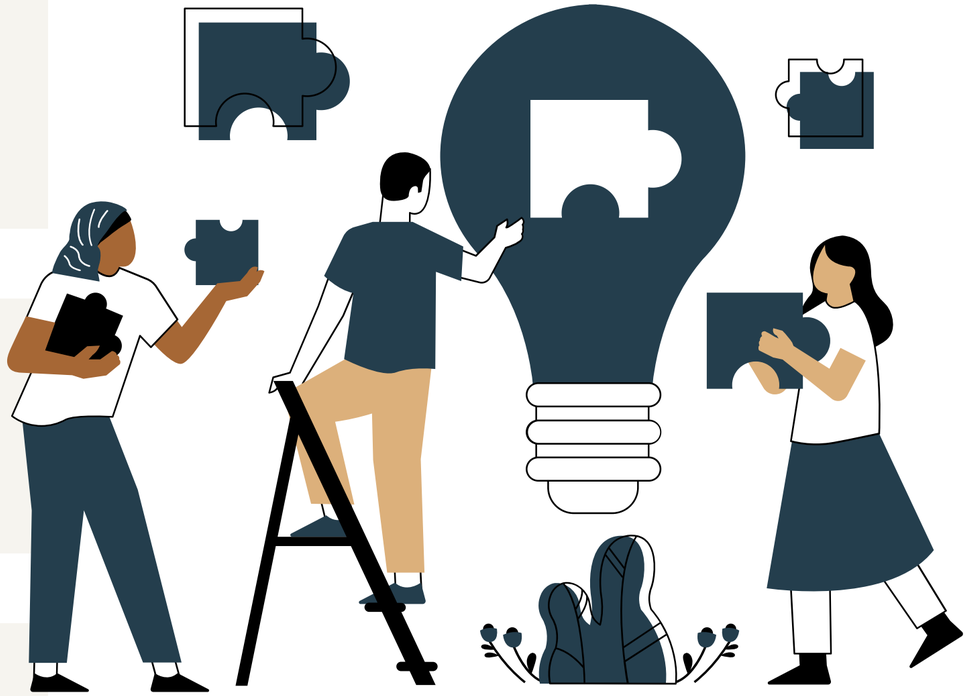
Trillions of BRL moved every month

Overnight operations are the linchpin of financial stability, balancing liquidity and regulatory compliance daily.

3

National Treasury of Brazil is making high efforts in tokenization of bonds and DREX

The National Treasury of Brazil is actively pursuing the tokenization of bonds and the development of DREX, generating an unique timing for those kinds of applications



However, the existing overnight system faces **significant operational and cost challenges!**

High Costs in Overnight Operations

Overnight operations in the financial market are costly, directly impacting the efficiency and profitability of institutions. An innovative solution is crucial to cut costs and improve operational margins.

Liquidity and Risk Management Challenges

Effective liquidity and risk management in overnight operations are ongoing challenges, requiring advanced solutions to minimize volatility and ensure financial stability for involved institutions.

Avalanche Overnight: the **best** solution

We have developed a revolutionary Overnight System using the Avalanche, aiming to disrupt the market and establish a new standard for Brazil's financial institutions.

1. Brazilian Treasury Bonds Tokenization

Complete Smart Contract and Blockchain infrastructure for operating Tokenized Treasury Bonds inside Avalanche.

2. Amazing and simple User Experience

Complete Smart Contract and Blockchain infrastructure for operating Tokenized Treasury Bonds inside Avalanche.

3. On top of the best benefits of AVAX

Our system offers the best Avalanche can give. Speed, transparency, and cost-effective transactions!

Watch our Demo

Avalanche Overnight

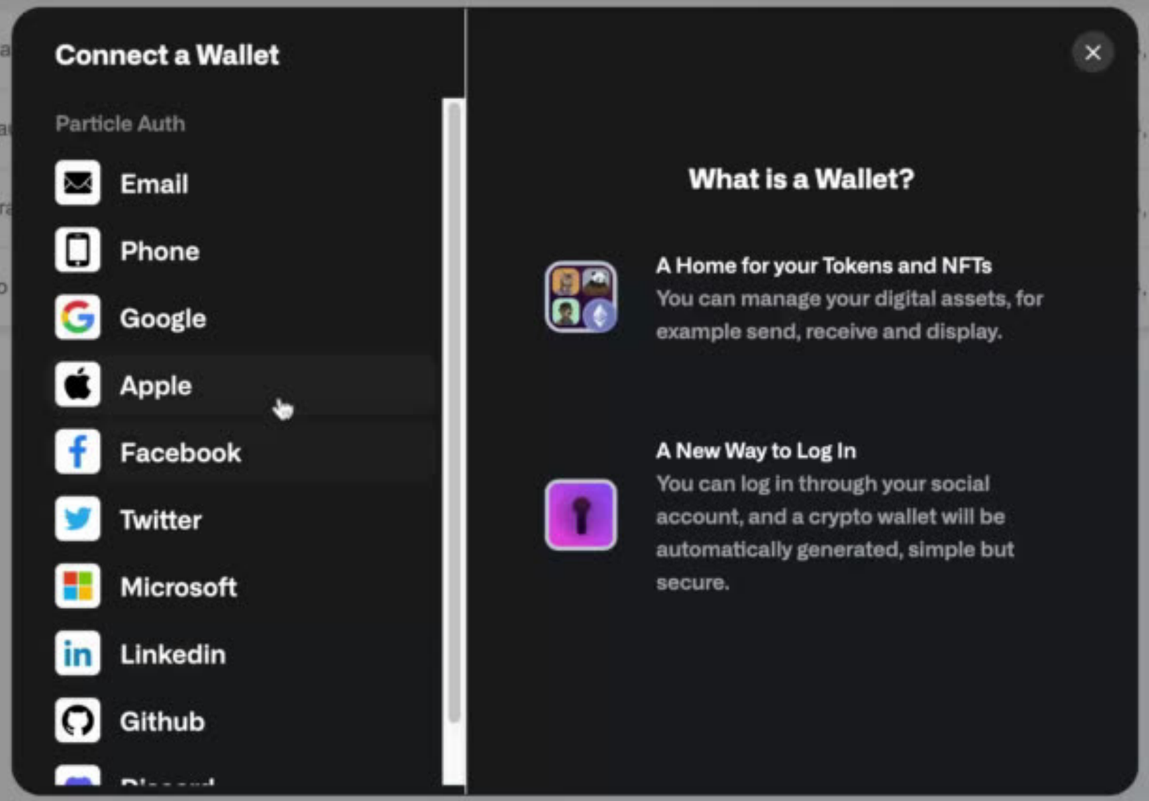
- Home
- My Dashboard

Logout

Overnight operations menu

Apply for credit

INSTITUTION	AMOUNT	BOND FOR COLLATERAL	DATE/TIME	STATUS
Banco Ca			, 00:50:00	R$ 0,00 / R$ 10.000,00
Banco Ita			, 00:40:00	R$ 0,00 / R$ 20.000,00
Banco Bra			, 00:30:00	R$ 0,00 / R$ 30.000,00
Banco do			, 00:20:00	R$ 0,00 / R$ 10.000,00



GitHub Repo

01. Create Liquidity Request

Call this function if you need money for reserves which the Central Banks asks for.

```
function createLiquidityRequest(string memory _institution, uint256 _totalAmount, address _collateralAsset) public onlyPrivileged returns(bool) {
    // Create a new instance of liquidityRequest directly in storage
    liquidityRequest storage newRequest = liquidityRequests.push();

    // Define the properties of the new liquidity request
    newRequest.institution = _institution;
    newRequest.institutionAddress = msg.sender;
    newRequest.totalAmount = _totalAmount;
    newRequest.raisedAmount = 0;
    newRequest.collateralAsset = _collateralAsset;
    newRequest.collateralAmount = (_totalAmount * 10 ** 18) / (INTBt(_collateralAsset).getTokenPrice());
    newRequest.requestDate = block.timestamp;
    newRequest.status = Status.Open;

    // Lock collateral assets on smartcontract
    privilegedTransferNTBt(_collateralAsset, msg.sender, address(this), newRequest.collateralAmount);

    return true;
}
```

02. Provide Liquidity

Provide liquidity for other banks, maintain great relations and ensure the functioning of financial system

```
function provideLiquidity(uint256 _liquidityRequestIndex, uint256 _provideAmount) public onlyPrivileged returns (bool) {
    //Check if the liquidity request exists
    require(_liquidityRequestIndex < liquidityRequests.length, "Request index out of bounds");

    //Checks whether the amount sent exceeds the liquidity request
    require(_provideAmount <= (liquidityRequests[_liquidityRequestIndex].totalAmount - liquidityRequests[_liquidityRequestIndex].raisedAmount),

    //Send money to the bank and register it in the system
    privilegedTransferReal(msg.sender, liquidityRequests[_liquidityRequestIndex].institutionAddress, _provideAmount);
    liquidityRequests[_liquidityRequestIndex].raisedAmount += _provideAmount;

    //Verify if the provider is already on the liquidity providers list and add him
    if (liquidityRequests[_liquidityRequestIndex].liquidityProviders[msg.sender] == 0) {
        liquidityRequests[_liquidityRequestIndex].liquidityProviderAddresses.push(msg.sender);
    }

    liquidityRequests[_liquidityRequestIndex].liquidityProviders[msg.sender] += _provideAmount;

    //Increase his Liquidity Score
    liquidityProvidersScore[msg.sender] += _provideAmount;

    return true;
}
```

03. Pay liquidity providers

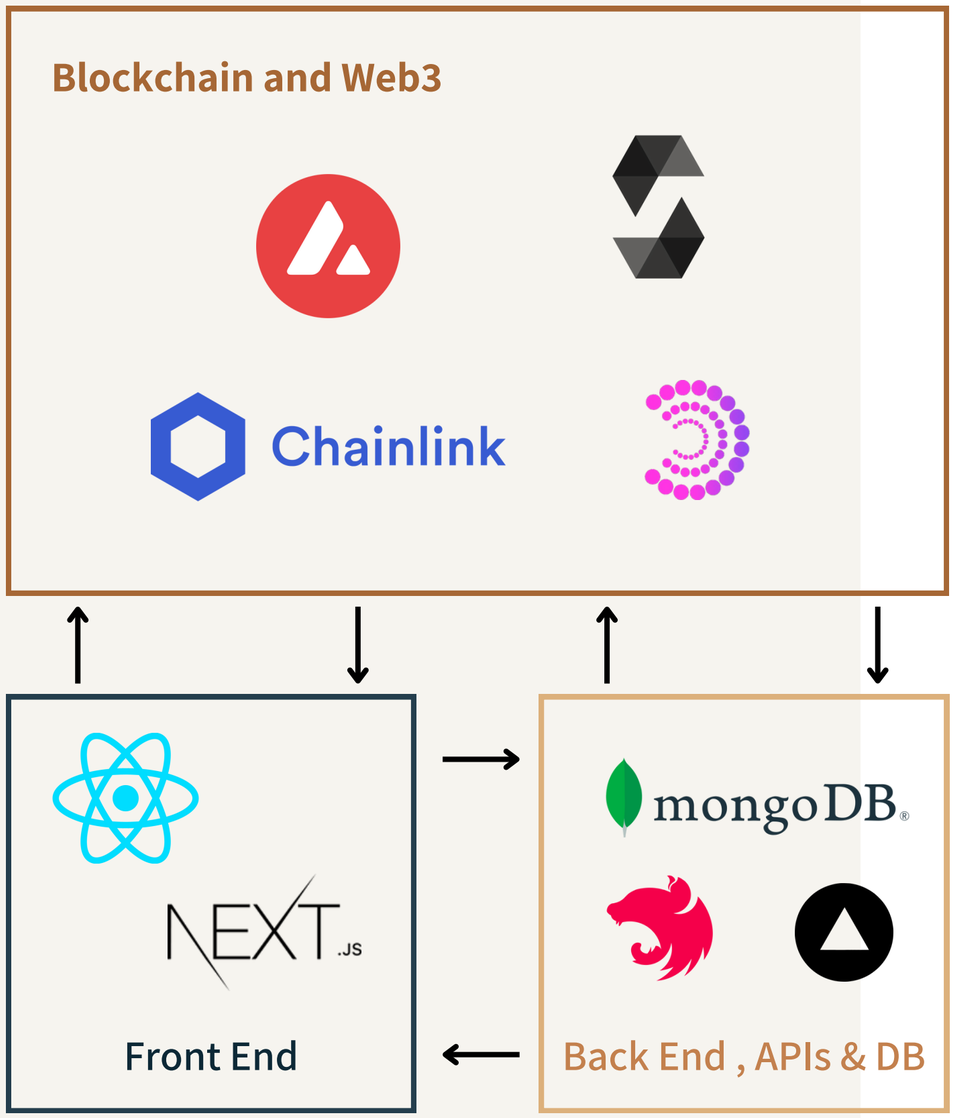
After the overnight, payback the banks who provided liquidity for your request

```
function payCreditors(uint256 _liquidityRequestIndex) public onlyPrivileged returns (bool) {
    require(_liquidityRequestIndex < liquidityRequests.length, "Request index out of bounds");
    liquidityRequest storage request = liquidityRequests[_liquidityRequestIndex];

    require(msg.sender == request.institutionAddress, "This request is not yours!");

    //Pay all the ones who provided liquidity with a daily interest rate (In Brazil, it is called Selic Over)
    for (uint i = 0; i < request.liquidityProviderAddresses.length; i++) {
        address provider = request.liquidityProviderAddresses[i];
        uint256 amount = request.liquidityProviders[provider];

        if (amount > 0) {
            uint256 realAmount = (amount * INTBt(request.collateralAsset).getTokenPrice())/(10**18);

            privilegedTransferReal(msg.sender, provider, realAmount);
            // Resetar o montante para evitar re-pagamentos
            request.liquidityProviders[provider] = 0;
        }
    }

    //Send the collateral back and close the request
    privilegedTransferNTBt(request.collateralAsset, address(this), request.institutionAddress, request.collateralAmount);
    request.status = Status.Closed;

    return true;
}
```

04. Default payment

Pay collateral in case the bank defaults.

```
function defaultPayment(uint256 _liquidityRequestIndex) public onlyPrivileged returns (bool) {
    // Ensure the provided request index is within bounds
    require(_liquidityRequestIndex < liquidityRequests.length, "Request index out of bounds");
    liquidityRequest storage request = liquidityRequests[_liquidityRequestIndex];

    // Ensure that the request status is Open, allowing for default
    require(request.status == Status.Open, "Status closed");

    // Check if 24 hours have passed since the requestDate
    require(block.timestamp >= (request.requestDate + 86400), "Cannot default before 24 hours");

    uint256 amountProvided = request.liquidityProviders[msg.sender];
    require(amountProvided > 0, "No liquidity provided by sender");

    // Calculate the proportion of collateral to be returned based on the provided amount
    uint256 collateralToReturn = (amountProvided * 10 ** 18 / INTBt(request.collateralAsset).getTokenPrice());

    // Transfer the proportional collateral back to the sender
    privilegedTransferNTBt(request.collateralAsset, address(this), msg.sender, collateralToReturn);

    // Update the raisedAmount and collateralAmount in the request
    request.raisedAmount -= amountProvided;
    request.collateralAmount -= collateralToReturn;

    // Remove the liquidity contribution from the provider
    request.liquidityProviders[msg.sender] = 0;

    return true;
}
```



Blueprint of Technologies

We have developed a complete application and MVP, using the AVAX potential to create a scalable solution for the clients.

Market Opportunity

R$ 1,1 trillion per month

TOTAL MARKET

99.6%

% of total repurchase operations

9.186

OVERNIGHT OPERATIONS per day

In managing short-term liquidity, the BCB borrowed overnight funds on all business days of the month. The average financial volume of these borrowing operations reached R$1.1 trillion at a maximum rate of 12.65% p.y. until 11/1 and of

Future Roadmap

What are we looking forward to achieve with Avalanche Overnight.

Technical Development: Finish the Beta version and correct errors

Strategic Partnerships: Know-how and financial institutions partnerships

Short-term

New features release: Multiple Owner Account / Multisig

Go-to-Market Strategy: Acquire clients and improve considering MVP feedbacks

Mid-term

New operations types: Keep improving the finantial markets with Avalanche

Cross-border Solution: Explore new markets and countries for our product

Long-term

Thank You!

GitHub Repo

https://github.com/ryanviana/avalanche-overnight