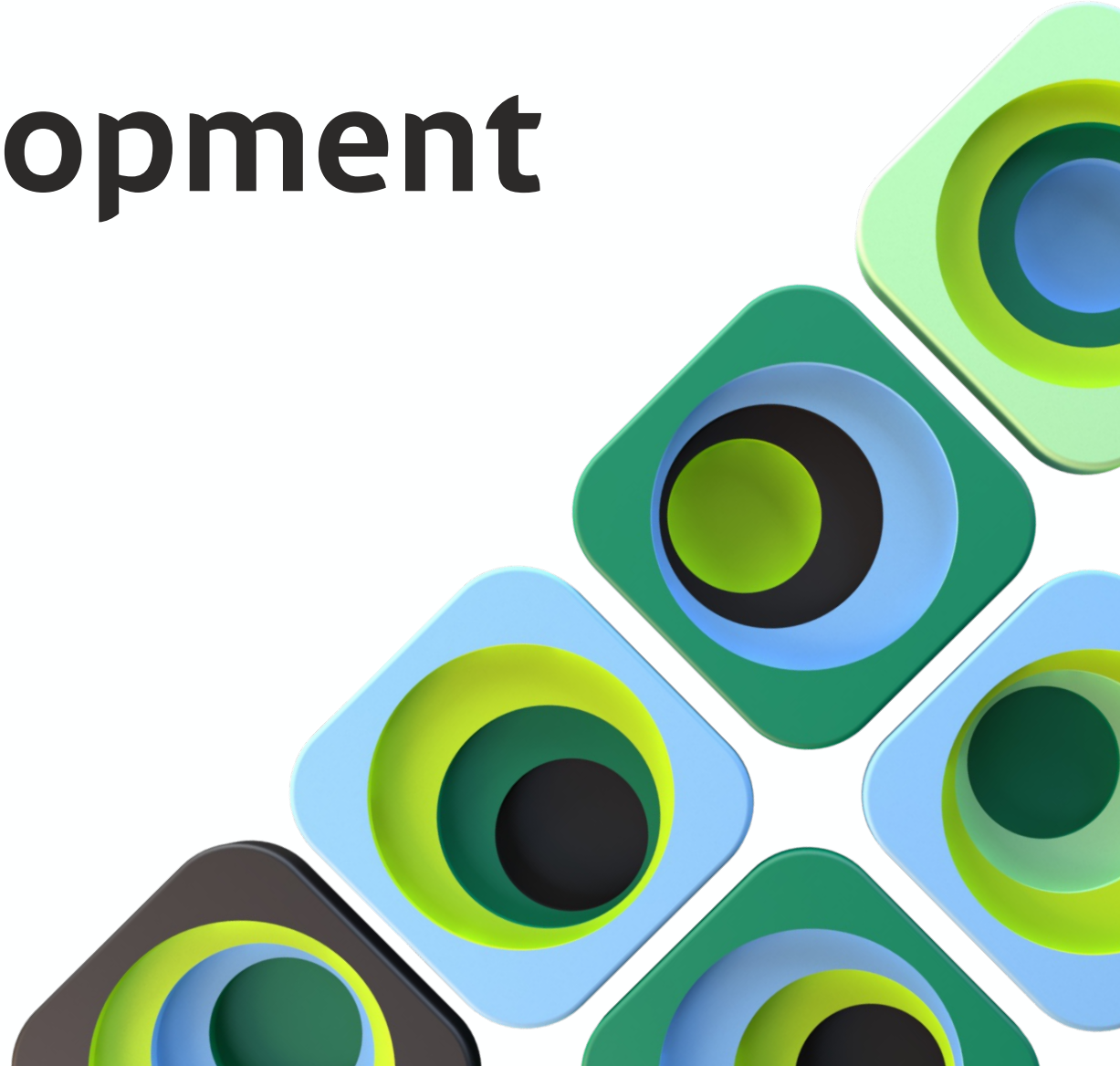
FullStack Development Test

Software Engineering

Technology



Wompi

Last rev. 9/Oct/2025 *by,*

Javier Alejandro Perez javier.perez@wompi.com,

Laura Ramirez laura.ramirez@wompi.co

The story

This test is intended to evaluate various aspects of front-end & back-end development, like code clarity, API / endpoints definition, design simplicity, test coverage or correct use of the HTTP protocol, in a real-life like scenario, UI development, data management, infrastructure as a code, software development principles and use of AI as coding assistant.

In this test you must create an **App for a Product paid with Wompi.** It is the onboarding where we obtain customer payment data to make the transaction of specific product, obtain customer delivery data, show the result of the payment process, and update stock data[1].

General business process

1. You must create a UI or page to show the product and units available in the stock from your store (with description and price), that will be bought by the customer.

2. Show to the customer the payment type button "*Pay with credit card*". That opens a Modal requesting Credit Card info.

3. Customer will complete Credit Card data and must be validated, (MasterCard and VISA number card detection logos is a plus) and Delivery information. *(Remember, credit card data must be fake but must follows the structure of credit cards),*

4. Next show the summary payment that includes (Product amount, Base fee added always and Delivery Fee) with payment button in a backdrop component[2].

5. When user click payment button:
    1. Create a transaction in PENDING in your backend API and obtain a transaction number.
    2. Call Wompi API to complete the payment.
    3. Once payment is completed or failed:
        1. Update the transaction with the transaction result in your Backend
        2. Assign the product that will be delivered to the customer
        3. Update the stock of your product.

6. Finally show the result of the transaction made and redirect to the product page with the stock updated.

Wompi

References

1. https://soporte.wompi.co/hc/es-419/articles/360057781394--C%C3%B3mo-pagar-con-tarjetas-a-trav%C3%A9s-de-Wompi-
2. https://m2.material.io/components/backdrop

The story

Your responsibilities

- ✓ The overall design of the API and information architecture (DB schema, folder structure, etc.)
- ✓ Deciding what data should be requested/responded on each endpoint. (*Add Postman Collection or Swagger URL public in README.md*)
- ✓ Deciding what validations each endpoint should perform, thinking of real-life situations that could arise.
- ✓ Safely handling sensitive data.
- ✓ The API you create must have stock, transactions, customers and deliveries. They must perform different types of requests.
- ✓ You must create a UI or page to show the product and units available in the stock from your store.
- ✓ Your app must be resilient; the App should recover the progress made by the client in case of refresh.
- ✓ Attention to detail is key.
- ✓ This App follows 5-step screen business process.

1. Product page → 2. Credit Card/Delivery info → 3. Summary → 4. Final status → 5. Product page

Wompi

References

1. https://soporte.wompi.co/hc/es-419/articles/360057781394--C%C3%B3mo-pagar-con-tarjetas-a-trav%C3%A9s-de-Wompi-
2. https://m2.material.io/components/backdrop

Development

1. **Frontend:** You must implement a single page app (SPA) built in either **ReactJS** or **VueJS**. NO other frameworks are allowed.

    1. The **App must be mobile oriented**, and you as a candidate create a Front that will manage multiple screen sizes. Making your App responsive but focus on mobile design. (Minimum screen size reference: iPhone SE (2020) 1334 x 750 pixels). UI interaction will work properly and must fit into UI boundaries.
    2. Use of either Redux or Vuex **is mandatory,** following Flux Architecture alignments as possible. Additionally, store the payment transaction data securely in the *application state* or *localStorage.*
    3. UX Design could be created as your own consideration.
    4. Use CSS frameworks of your preference, but we foster you to use flexbox or grid APIs[1].

2. **Backend:** You must implement an API Backend services built in either **JavaScript/TypeScript** or **Ruby**. With **Nest.js**, **Grappe** or **Sinatra**. Ruby on rails and other frameworks are NOT allowed.
    1. Business logic must NOT be handled within the routing / controller layer, could be on a separate layer. Try Hexagonal Architecture and Ports & Adapters pattern.
    2. Try to use Railway Oriented Programming (ROP), for your Use Cases.
    3. You can use any type of database. We recommend using **PostgreSQL** or **DynamoDB**. (*Add Data model design in README.md*)
    4. You can use any type of **ORM** and serialization library.
    5. The database must be seeded with dummy Products. There's NO need for an endpoint to create new products.

3. **Unit tests is mandatory either Frontend and Backend** for the competition of the tests, more than 80% of the coverage. (*Add the results in README.md and create them with Jest*)

4. **Publish the app using any Cloud provider**, we recommend AWS[2] (*with the AWS free tier, AWS Lambda, ECS, EKS, CloudFront, S3, RDS or DynamoDB*).

Wompi

References

1. *https://css-tricks.com/snippets/css/a-guide-to-flexbox/*
2. *https://engineering.teknasyon.com/single-page-application-deployment-on-aws-4b53d08b3ad3*

Test tools (UAT)

1. Wompi API. For starters, you must read:
    1. https://docs.wompi.co/docs/colombia/inicio-rapido/
    2. https://docs.wompi.co/docs/colombia/ambientes-y-llaves/

2. Use the following information to use API Keys or to access to an account created for this test. *Important note: This user could be used by any people doing the same test and your user session could be closed by different location accessing at the same time, try to integrate with API Keys.*

    - Is important always the tests must be on **Sandbox mode.**
    - **Sandbox mode** is in the main menu → Developers
    - Wompi User **(DON'T MODIFY CREDENTIALS AND DON'T ADD 2MFA)**
        - URL: https://login.staging.wompi.dev/
        - User: smltrs00
        - Password: ChallengeWompi123*
    - API URLs
        - UAT_URL=https://api.co.uat.wompi.dev/v1
        - UAT_SANDBOX_URL=https://api-sandbox.co.uat.wompi.dev/v1
    - API Keys Sandbox
        - pub_stagtest_g2u0HQd3ZMh05hsSgTS2lUV8t3s4mOt7
        - prv_stagtest_5i0ZGlGiFcDQifYsXxvsny7Y37tKqFWg
        - stagtest_events_2PDUmhMywUkvb1LvxYnayFbmofT7w39N
        - stagtest_integrity_nAlBuqayW70XpUqJS4qf4STYilSd89Fp

3. Make sure you use the Sandbox environment. That means, there's NO need for any real money transactions, in the context of this test.

4. Try to use AI CLI Assist to make your integrations.

Considerations

✅ We recommend create branches and pull requests by feature.
✅ We recommend the use of AI if possible.
⚠️ You must push your code to GitHub as a public repository only!, and please do NOT use words like "Wompi" (company name), in your repository, and do not share your solution with other developers or candidates.

References

1. *https://css-tricks.com/snippets/css/a-guide-to-flexbox/*
2. *https://engineering.teknasyon.com/single-page-application-deployment-on-aws-4b53d08b3ad3*

Wompi

Deliverables

1. Completion of Frontend App and Backend API
2. Link of GitHub repository with README.md updated.
3. Link of the AWS App deployed working connected with your Backend API.

Evaluation rubrics

1. **[5 points]** for README completed correctly.
2. **[5 points]** Images that renders fast and avoid UI/UX Out-boundaries.
3. **[20 points]** Full functionality of the onboarding process for Credit card payment checkout.
4. **[20 points]** for API working correctly.
5. **[30 points]** for more than 80% of coverage with Unit testing Backend and Frontend
6. **[20 points]** for App and API deployed through any Cloud provider.

---

1. **[5 *Bonus points*]** use of OWASP alignments, use of HTTPS and Security headers.
2. **[5 *Bonus points*]** for Apps completely responsive and working through different browsers.
3. **[10 *Bonus points*]** CSS skills.
4. **[10 *Bonus points*]** Clean code.
5. **[10 *Bonus points*]** for structuring your app using Hexagonal Architecture with Ports & Adapters pattern.
6. **[10 *Bonus points*]** for following ROP.

Considerations

✅ To complete the test, you need at least 100 points.
✅ If you obtain the minimum points will continue with the interview step.
⚠️ If it is detected that the repository does not have progress or commits, or that it is like the solution of another candidate, the test will be automatically voided for fraud.

Wompi

References

1. *https://observatory.mozilla.org/*

Try your best, candidate!

#EfectoWompi