The challenge: a minimalistic Badi app

We want you to build a simplified Badi app. The platform should focused on room search with three components:

- Address search; a screen where the user inputs the search term and see as result addresses and/or cities where we have rooms.
  We expect to have a custom service of forward geocoding. You can play a bit with [OpenCage]() to see the kind of service that we expect, of course, a simpler version.
  We would like to show results as soon as the user starts typing and not requiring them to press a button to perform the search. Bear in mind that with this approach there could be a number of optimisations, in client and backend side, to avoid spamming our servers while still providing smooth user experience.

- Whenever the user clicks in one result, the app should navigate to a screen where all the rooms that match with that search are shown.
  Consider adding:
  - Search within the selected location's boundaries
  - Smooth transitions while navigating the app
  - Custom filters for the search results
  - Layout flexibility; ie display results differently depending on device and/or orientation
  - Show rooms relevant data (at least some location information, image, price, popularity based on how many times the room has been seen, etc.)
  - Seamless pagination -- infinite scrolling -- whenever a user reaches the latest element.

- When clicking in a room, we expect a different transition to a new screen with detailed room information. In iOS, we would like you to use our [StackViewController]() and [StackModules](). The view needs to have different screen sections, well modularised.

- Gather some metrics and use them in a way that gives some benefits to the user. Some key events that we may want to record:
  - Typing in the search bar
  - Clicking in a location
  - Clicking in a room to navigate to the details

Deliverables

The project must be built in collaboration with your Ruby and iOS peers. At the end of the project, and before the presentation, you need to deliver the following

- Your git history and git flow will give us a better picture of your development process. Share private repos on GitHub with full access. Specifically
    - One Ruby repo,
    - One Swift repo, for iOS.
- At Badi we trust our Engineers to reason about their choices. Everyone in our team would need to be able to defend their opinions and decisions, ie: *why have you used a certain data structure, which alternatives did you consider, what would you improve if you had more time, etc*. We also know team make compromises that might result in different engineering decisions. Please justify properly all those. Those justifications should be included on a Readme file in each private repo.
- **(Ruby) We'd like to see the new backend deployed**. Our preference is AWS, but you can use other services like GCP, Heroku, Digital Ocean, etc.
- **Projects MUST build and run out of the box** without us having to do any further adjustments.

iOS guidelines

We want you to build an Xcode project from scratch while working in different aspects of the day to day operations @ Badi.

We value:

- Swifty code, our whole app is built in Swift, therefore this test must be written in the latest version of our beloved programming language.
- Project architecture: @ Badi you'll use VIPER so we encourage the use of it in your code challenge. What we do really want to see is a good project structure with clear responsibilities of each component.
- Code quality: appropriate data structures and the use of typical programming patterns and good practices.
- Documentation: while we don't document all our code we do want to have clear documentation of key, complex or reusable components of our projects.
- Testing: both having tests (Unit, Snapshot, UI, integration, and every kind of test you can imagine) and testable code.
- Extensibility: whether your decisions in the codebase are scalable and would still be valid in future iterations.

- UI: we make our UI with nibs so we would like to see you using them, keep in mind that it should resize properly in different screens (avoid fixed sizes).

Ruby guidelines

We want you to build on top of a small Ruby on Rails / Grape project (provided by us) and work on different aspects of the day to day operations a backend engineer must perform @ Badi.

We value the following:

- Project architecture: @ Badi you'll use Grape and service objects so we encourage the use of them in your code challenge. What we do really want to see is a good project structure with clear responsibilities for each class.
- Performance: consider always performance as a key factor. Can we make this query more efficient? Can we unblock requests faster and do some stuff in the background? Let's do it!
- Code quality: appropriate data structures and the use of typical programming patterns and best practices.
- Testing: both having tests (unit-testing, integration, and every kind of test you can imagine) and testable code.
- Extensibility: whether your decisions in the codebase are scalable and would still be valid in future iterations.