Melhorando o tratamento de erros do seu App

Lucas Santos - Software Engineer @ iFood

Apple Developer Academy 2017/18

Software engineer @ iFood Dez/2018

lucas1295santos

Agenda

Porque tratar erros

Melhorias para aplicar no seu projeto

Agenda

Porque tratar erros

Melhorias para aplicar no seu projeto

Porque tratar erros

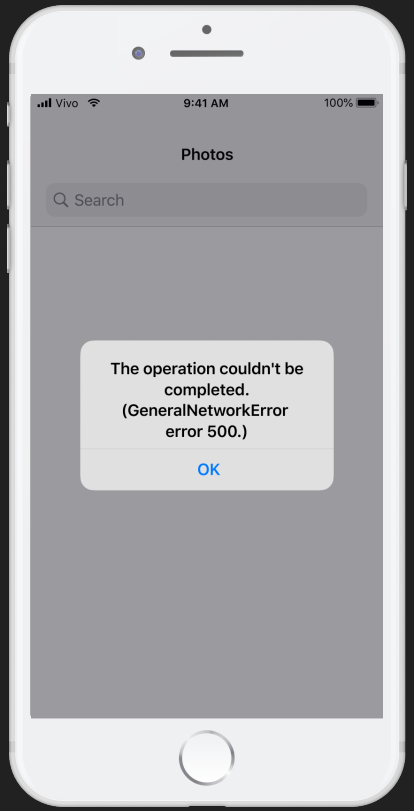
Descobrir problemas rápido

```
private func callMerchant(with telephoneNumber: String?) {
    guard let phone = telephoneNumber else { return }

    /*

     Calling a phone number code goes here

     */
}
```

Porque tratar erros

Descobrir problemas rápido

Melhorar a Experiência do Usuário (UX)


Vivo
9:41 AM
100%
Photos
Search
The operation couldn't be completed. (GeneralNetworkError error 500.)
OK

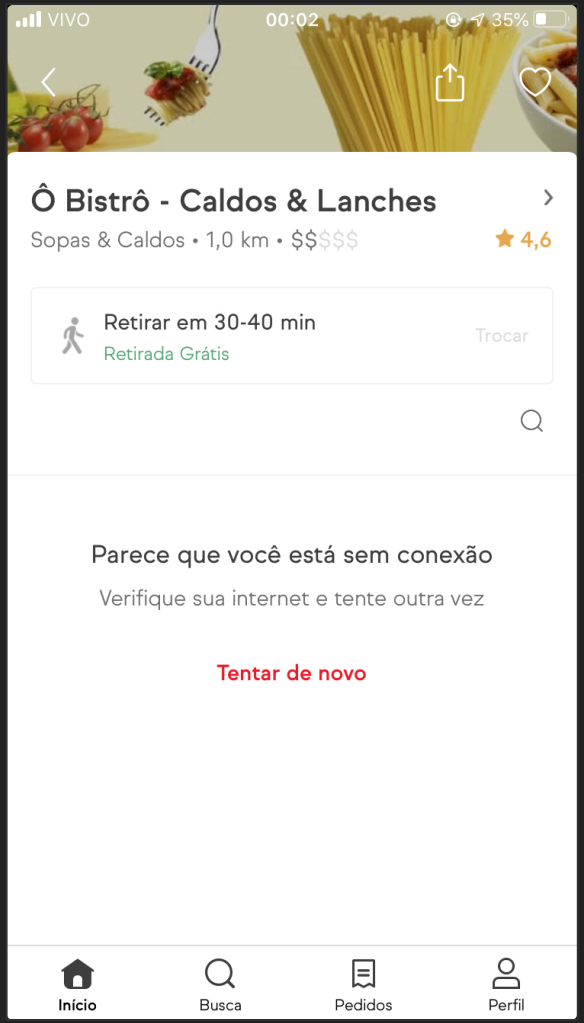

Search
23:58
38%
DETALHES DO ITEM
Pizza Paulista (25 cm diâmetro)
Massa com fermentação natural de 48h, molho
Este restaurante abre amanhã às
10:30
Mas você pode olhar o cardápio à vontade e voltar quando o restaurante estiver aberto.
Ok, entendi

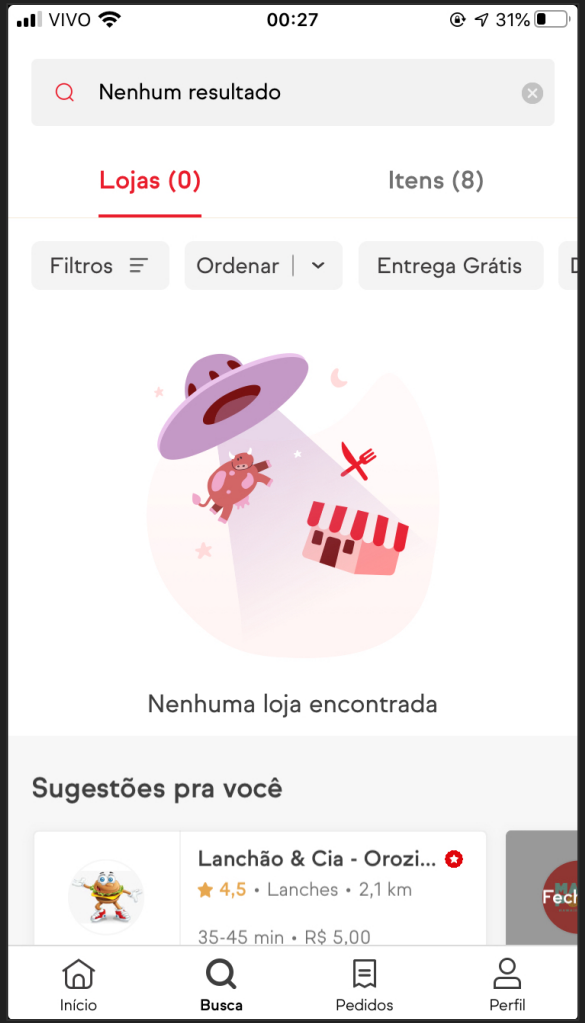
Porque tratar erros

Descobrir problemas rápido

Melhorar a Experiência do Usuário (UX)

Recuperação de um estado de erro


VIVO
00:02
35%
Ô Bistrô - Caldos & Lanches
Sopas & Caldos • 1,0 km • $$$$$
4,6
Retirar em 30-40 min
Retirada Grátis
Trocar
Parece que você está sem conexão
Verifique sua internet e tente outra vez
Tentar de novo
Início
Busca
Pedidos
Perfil


VIVO
00:27
31%
Nenhum resultado
Lojas (0)
Itens (8)
Filtros
Ordenar
Entrega Grátis
Nenhuma loja encontrada
Sugestões pra você
Lanchão & Cia - Orozi...
4,5 • Lanches • 2,1 km
35-45 min • R$ 5,00
Início
Busca
Pedidos
Perfil

Considere tratar erros sempre que...

🌎 Fizer requests

⌨️ Capturar input de usuário

⚙️ Usar algum Encode ou Decode

⏎ Escapar de uma função antes de sua total execução

Agenda

Porque tratar erros

Melhorias para aplicar no seu projeto

Melhorias para aplicar no seu projeto

Serviço de Log

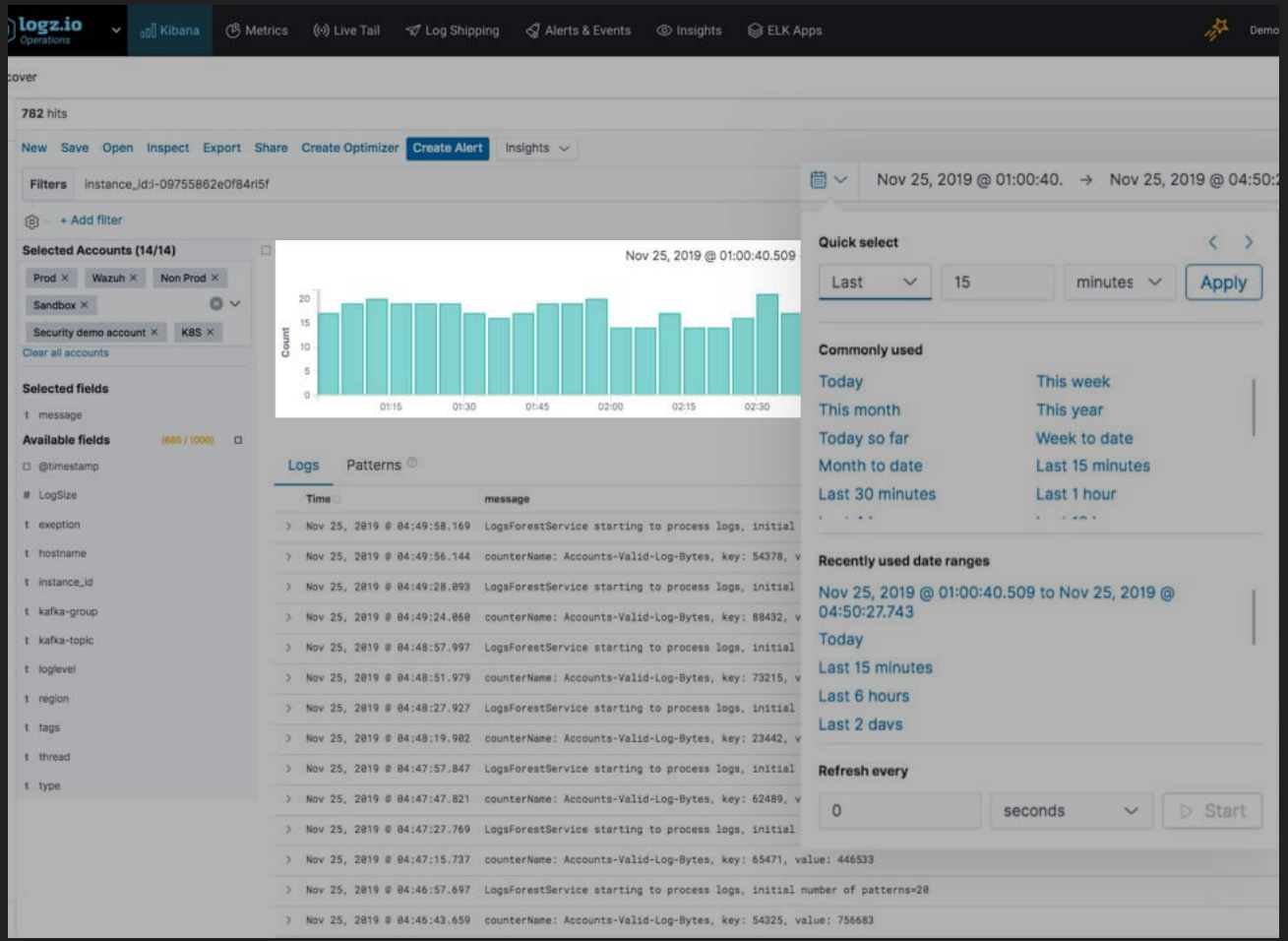
logz.io

Logs por data

Queries

Alertas no slack ou email

Dashboards

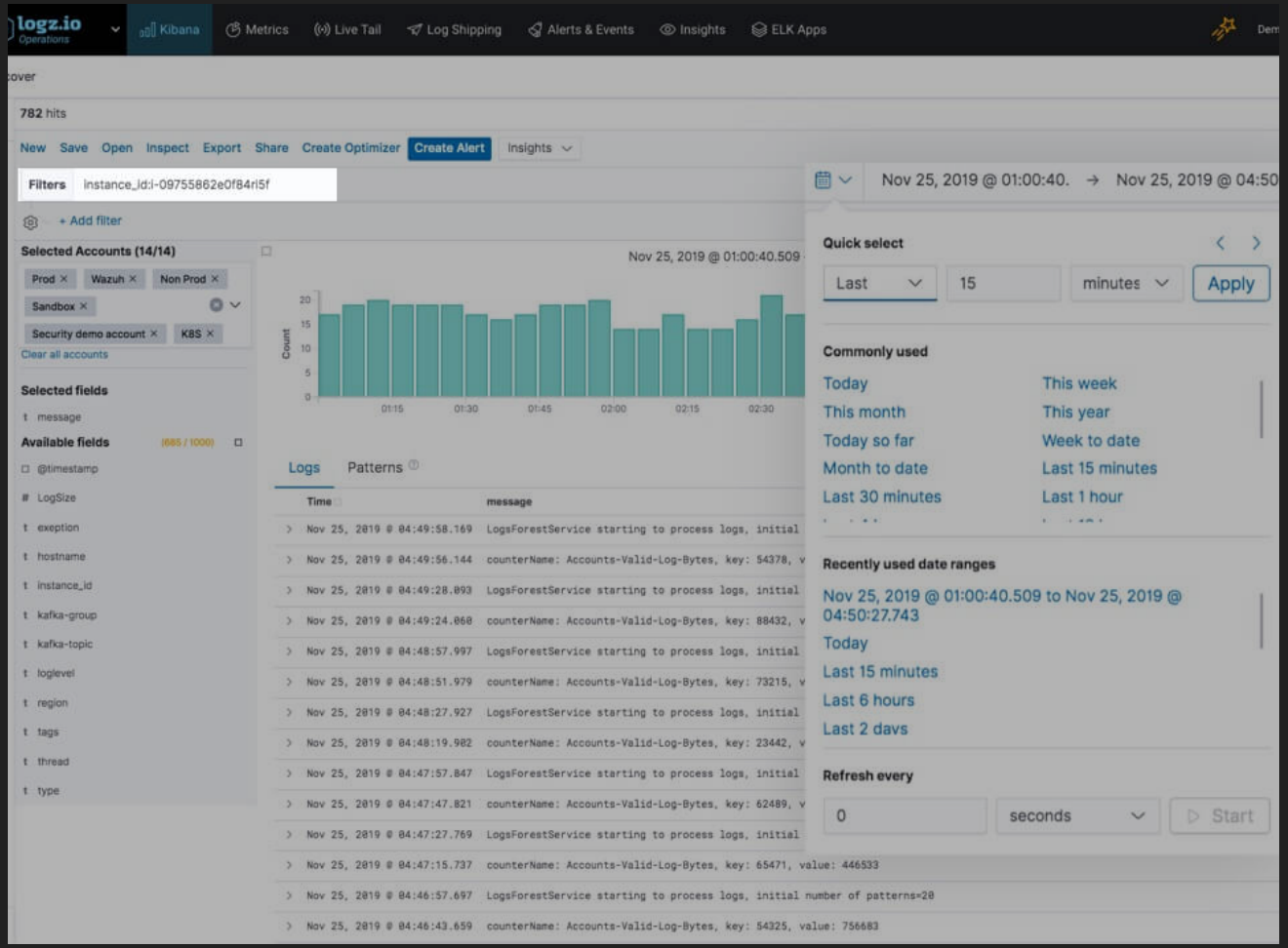


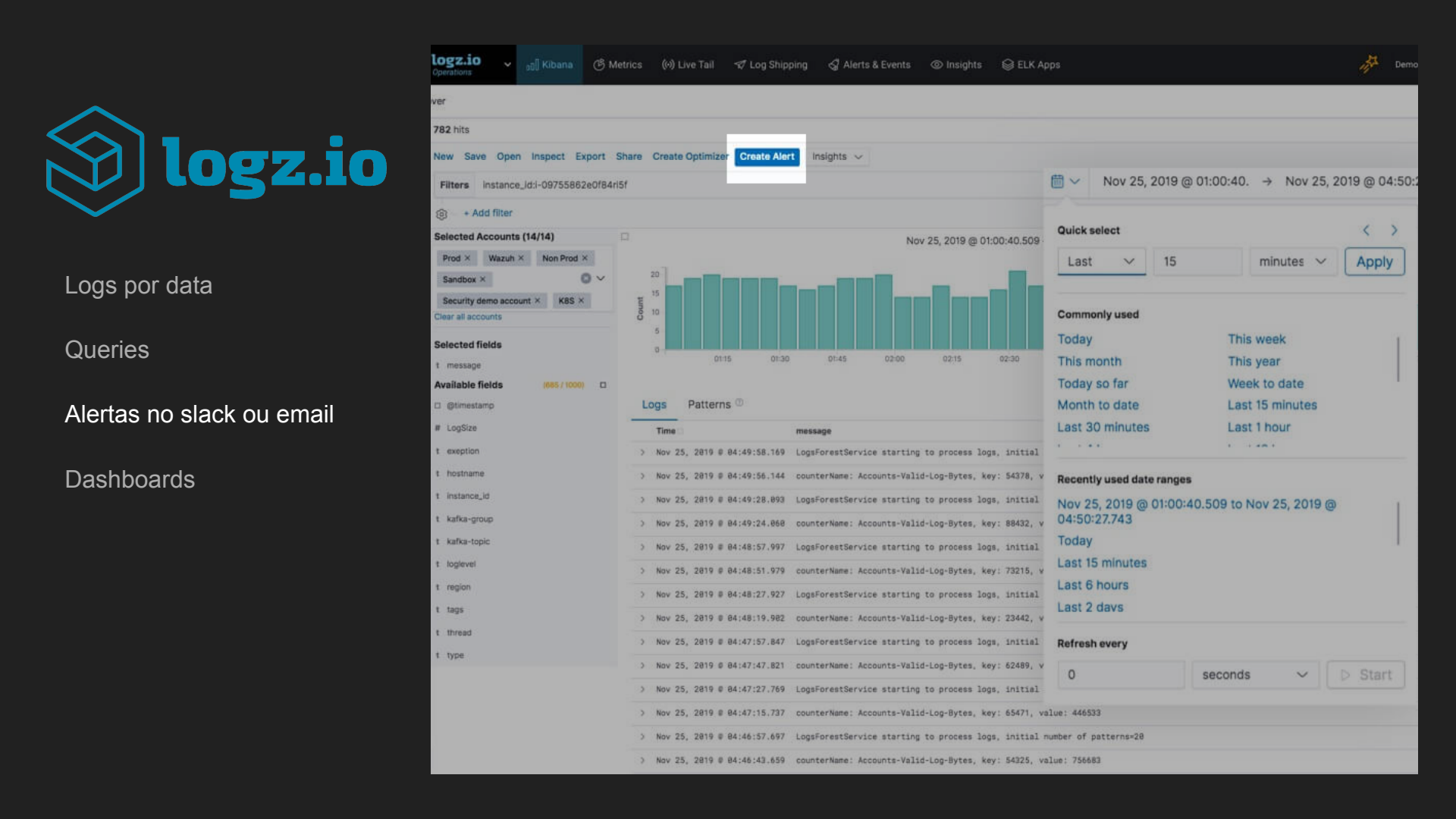
Logs por data

Queries

Alertas no slack ou email

Dashboards





logz.io

Logs por data

Queries

Alertas no slack ou email

Dashboards

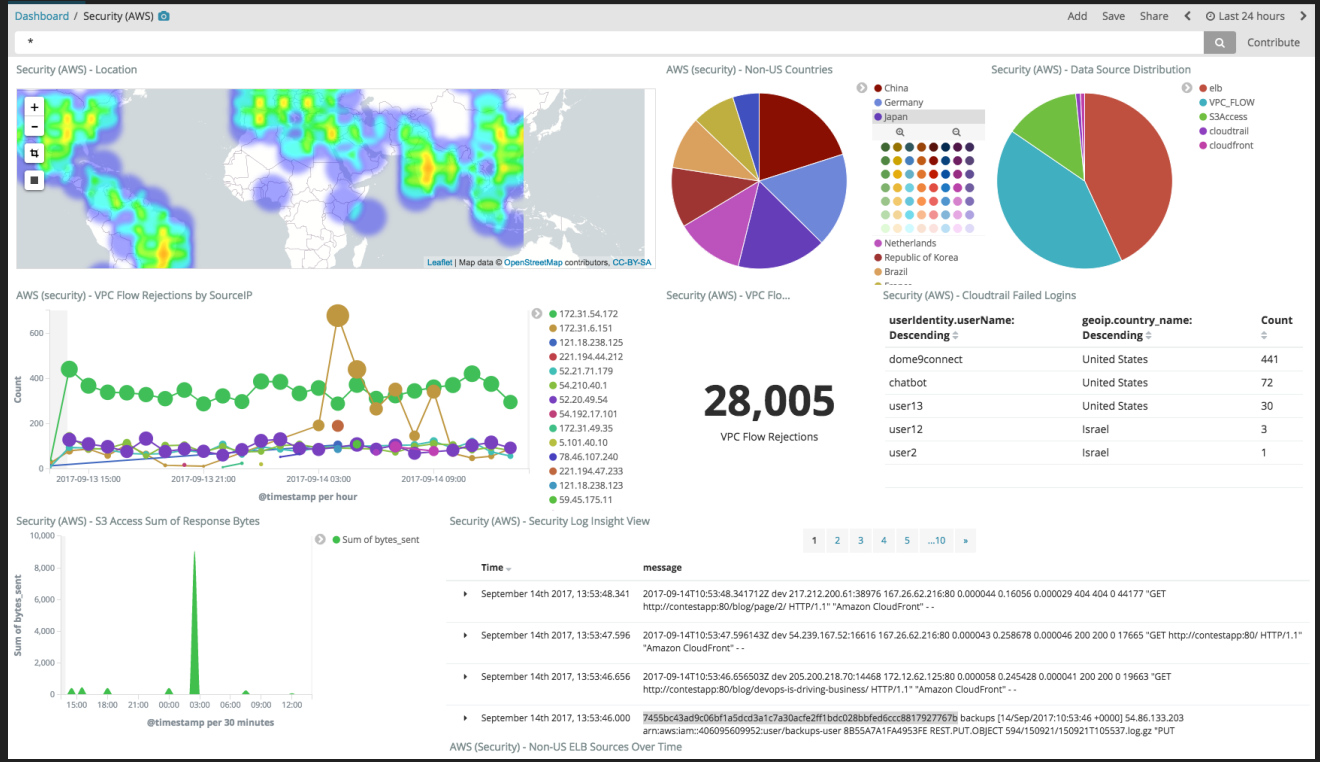
logz.io

Logs por data

Queries

Alertas no slack ou email

Dashboards



To do list de nova feature

- 🗺️ Mapear casos de erro
- 👨‍💻 Criar logs no código
- 🚨 Criar alertas paras os logs no logz.io
- 👀 Acompanhar os alertas

Melhorias para aplicar no seu projeto

Serviço de Log

Protocolo Error do Swift

```swift
enum SimpleError: Error {
    case generic
    case network(payload: [String: Any])
}

func simpleFunction(payload: [String: Any]) throws {
    throw SimpleError.network(payload: payload)
}
```

```swift
struct ComplexError: Error {
    enum ErrorType {
        case one
        case two
    }

    let line:Int
    let file: String
    let type: ErrorType
    let isUserLoggedIn: Bool
}

func complexFunction() throws {
    throw ComplexError(line: 187, file: "Main.swift", type: .one, isUserLoggedIn: false)
}
```

```swift
enum RegisterUserError: Error {
    case emptyName
    case invalidEmail
    case invalidPassword
}

extension RegisterUserError: LocalizedError {
    var errorDescription: String? {
        switch self {
        case .emptyName:
            return "Name can't be empty"
        case .invalidEmail:
            return "Not an email format"
        case .invalidPassword:
            return "The password must be at least 8 characters long, contain one digit and one capital letter"
        }
    }
}
```

Melhorias para aplicar no seu projeto

Serviço de Log

Protocolo Error do Swift

Não usar nil como erro

Retornar nulo 👎

```swift
func getUserPreferences() -> UserPreferences? {
    let dataFromKey = UserDefaults.standard.data(forKey: "user_preferences")
    guard let data = dataFromKey else {
        return nil
    }
    let decoder = JSONDecoder()
    let userPreferences = try? decoder.decode(UserPreferences.self, from: data)
    return userPreferences
}
```

Lançar uma exceção 👍

```
func getUserPreferences() throws -> UserPreferences {
    let dataFromKey = UserDefaults.standard.data(forKey: "user_preferences")
    guard let data = dataFromKey else {
        throw UserPreferencesError.noUserConfigStored
    }
    let decoder = JSONDecoder()
    let userPreferences = try decoder.decode(UserPreferences.self, from: data)
    return userPreferences
}
```

E como faz se for async?

Retornar nulo 👎

```
class MenuItem: Codable {
    let name: String
    let description: String
    let price: Float
}
```

Retornar nulo 👎

```
func fetchMenuItem(with id: String, completion: @escaping (MenuItem?) ->
Void) {
    Network.request(.menuItem(id: id)) { result in
        switch result {
        case .success(let data):
            let decoder = JSONDecoder()
            let menuItem = try? decoder.decode(MenuItem.self, from: data)
            completion(menuItem)
        case .failure:
            completion(nil)
        }
    }
}
```

Retornar nulo 👎

```
func fetchAndDisplayMenuItem(with id: String) {
    fetchMenuItem(with: id) { fetchedMenuItem in
        guard let menuItem = fetchedMenuItem else {
            return
        }
        displayMenuItem(menuItem)
    }
}
```

Passar um Result 👍

```
func fetchMenuItem(with id: String, completion: @escaping (Result<MenuItem, Error>) -> Void) {
    Network.request(.menuItem(id: id)) { result in
        switch result {
        case .success(let data):
            do {
                let decoder = JSONDecoder()
                let menuItem = try decoder.decode(MenuItem.self, from: data)
                completion(.success(menuItem))
            } catch {
                completion(.failure(error))
            }
        case .failure(let error):
            completion(.failure(error))
        }
    }
}
```

Passar um Result 👍

```
func fetchAndDisplayMenuItem(with id: String) {
    fetchMenuItem(with: id) { menuItemResult in
        switch menuItemResult {
        case .success(let menuItem):
            displayMenuItem(menuItem)
        case .failure(let error):
            Log.log(error)
            displayEmptyState(reason: .network)
        }
    }
}
```

Melhorias para aplicar no seu projeto

Serviço de Log

Protocolo Error do Swift

Não usar nil como erro

Separar código feliz do tratamento de erro

```
struct User {
    let name: String
    let email: String
    let password: String
}
```

```
do {
    try registerUser(user)
} catch {
    MyLoggerWrapperClass.log(error: error)
}
```

👎

Código relevante no fim

Violação à S.R.P.

```
func registerUser(_ user: User) throws {
    guard user.name.isEmpty == false else {
        throw RegisterUserError.emptyName
    }
    guard isValid(email: user.email) else {
        throw RegisterUserError.invalidEmail
    }
    guard isValid(password: user.password) else {
        throw RegisterUserError.invalidPassword
    }

    /*
        Registering an user code goes here
     */
}
```

Código relevante no topo

Conformidade à S.R.P.

```
func registerUser(_ user: User) throws {
    do {
        try validateUser(user)

        /*
            Registering an user code goes here
         */
    }
    catch {
        throw error
    }
}
```

```
func validateUser(_ user: User) throws {
    if user.name.isEmpty {
        throw RegisterUserError.emptyName
    } else if isValid(email: user.email) == false {
        throw RegisterUserError.invalidEmail
    } else if isValid(password: user.password) == false {
        throw RegisterUserError.invalidPassword
    }
}
```

Recapitulando

- 😌 Não deixe de tratar erros, seus usuários agradecem
- ☁️ Use uma ferramenta de log no seu projeto
- ❤️ Use o protocolo Error para ter erros expressivos
- 🤷‍♂️ Não use nil como erro
- 👨‍💻 Separe o código de validação e tratamento de erros da parte realmente importante

```
func thankPeople() {
  print("Obrigado 😄 ")
}
```

Slides, referências e links relevantes