

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

Por que tratar erros?

Melhorias para aplicar no seu projeto

Agenda

Por que tratar erros?

Melhorias para aplicar no seu projeto

Por que tratar erros

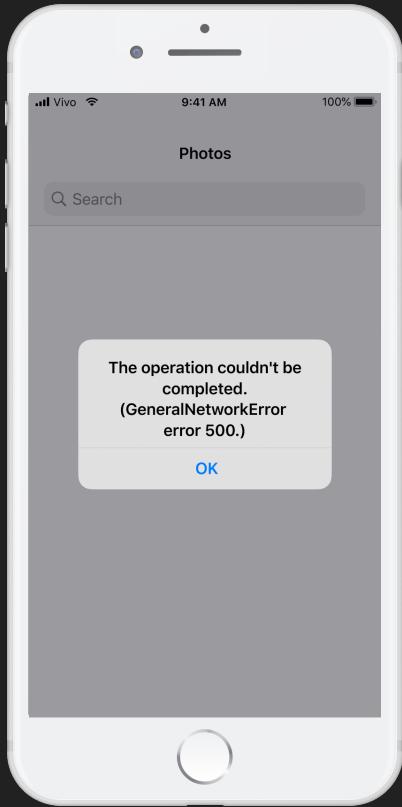
Descobrir problemas rápido

```
private func callMerchant(with telephoneNumber: String?) {  
    guard let phone = telephoneNumber else { return }  
  
    /*  
        Calling a phone number code goes here  
    */  
}
```

Por que tratar erros

Descobrir problemas rápido

Melhorar a Experiência do Usuário (UX)

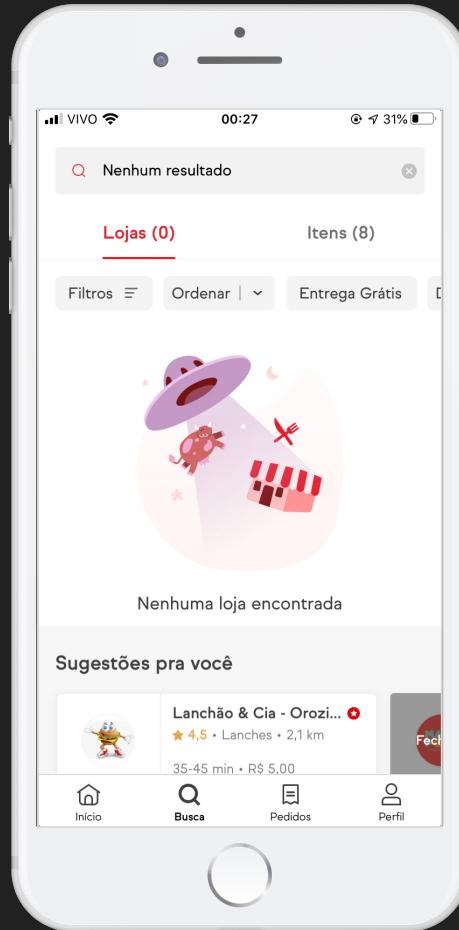
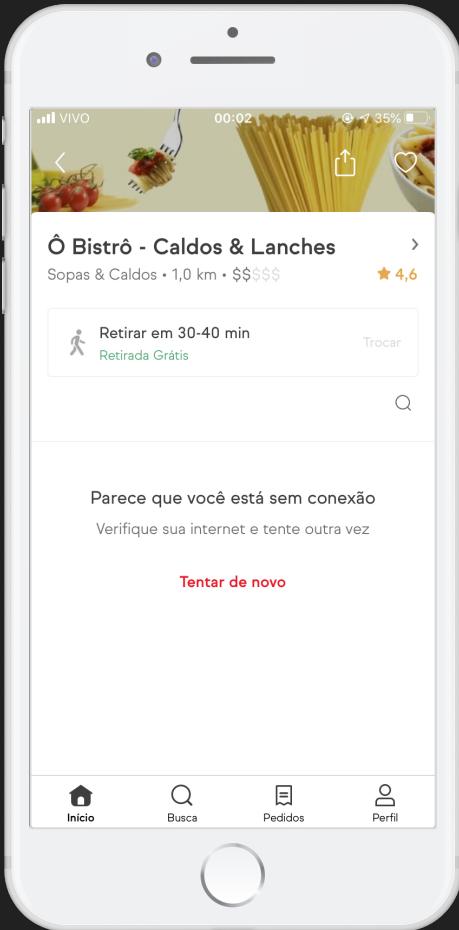


Por que tratar erros?

Descobrir problemas rápido

Melhorar a Experiência do Usuário (UX)

Recuperação de um estado de erro



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

Por que tratar erros?

Melhorias para aplicar no seu projeto

Melhorias para aplicar no seu projeto

Ferramenta de monitoramento





Logs por data

Queries

Alertas no slack ou email

Dashboards

Screenshot of the logz.io interface showing a search results page for the query "cover".

Key UI elements visible:

- Top navigation bar:** Operations, Kibana, Metrics, Live Tail, Log Shipping, Alerts & Events, Insights, ELK Apps, Demo.
- Search bar:** cover, 782 hits.
- Toolbar:** New, Save, Open, Inspect, Export, Share, Create Optimizer, Create Alert, Insights.
- Filters:** instance_id:i-09755862e0f84rl5f, Selected Accounts (14/14) including Prod, Wazuh, Non Prod, Sandbox, Security demo account, KBS.
- Selected fields:** t message, Available fields: @timestamp, # LogSize, t exception, t hostname, t instance_id, t kafka-group, t kafka-topic, t loglevel, t region, t tags, t thread, t type.
- Time range:** Nov 25, 2019 @ 01:00:40.509 to Nov 25, 2019 @ 04:50:27.743.
- Quick select:** Last 15 minutes.
- Commonly used:** Today, This week, This month, This year, Today so far, Week to date, Month to date, Last 15 minutes, Last 30 minutes, Last 1 hour.
- Recently used date ranges:** Nov 25, 2019 @ 01:00:40.509 to Nov 25, 2019 @ 04:50:27.743, Today, Last 15 minutes, Last 6 hours, Last 2 days.
- Refresh every:** 0 seconds.
- Logs table:** Shows log entries with columns Time and message, such as "Nov 25, 2019 @ 04:49:58.169 LogsForestService starting to process logs, initial".

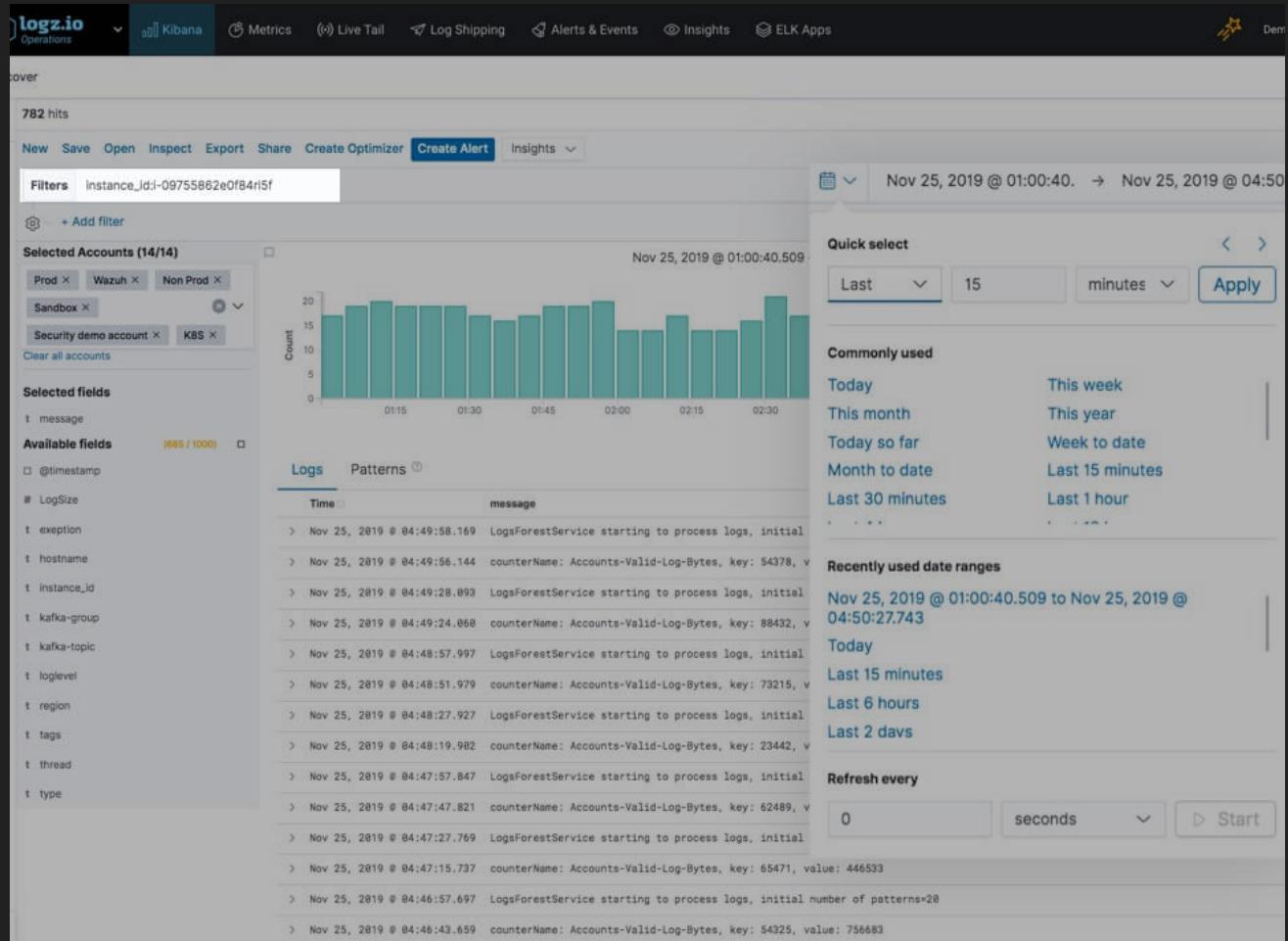


Logs por data

Queries

Alertas no slack ou email

Dashboards



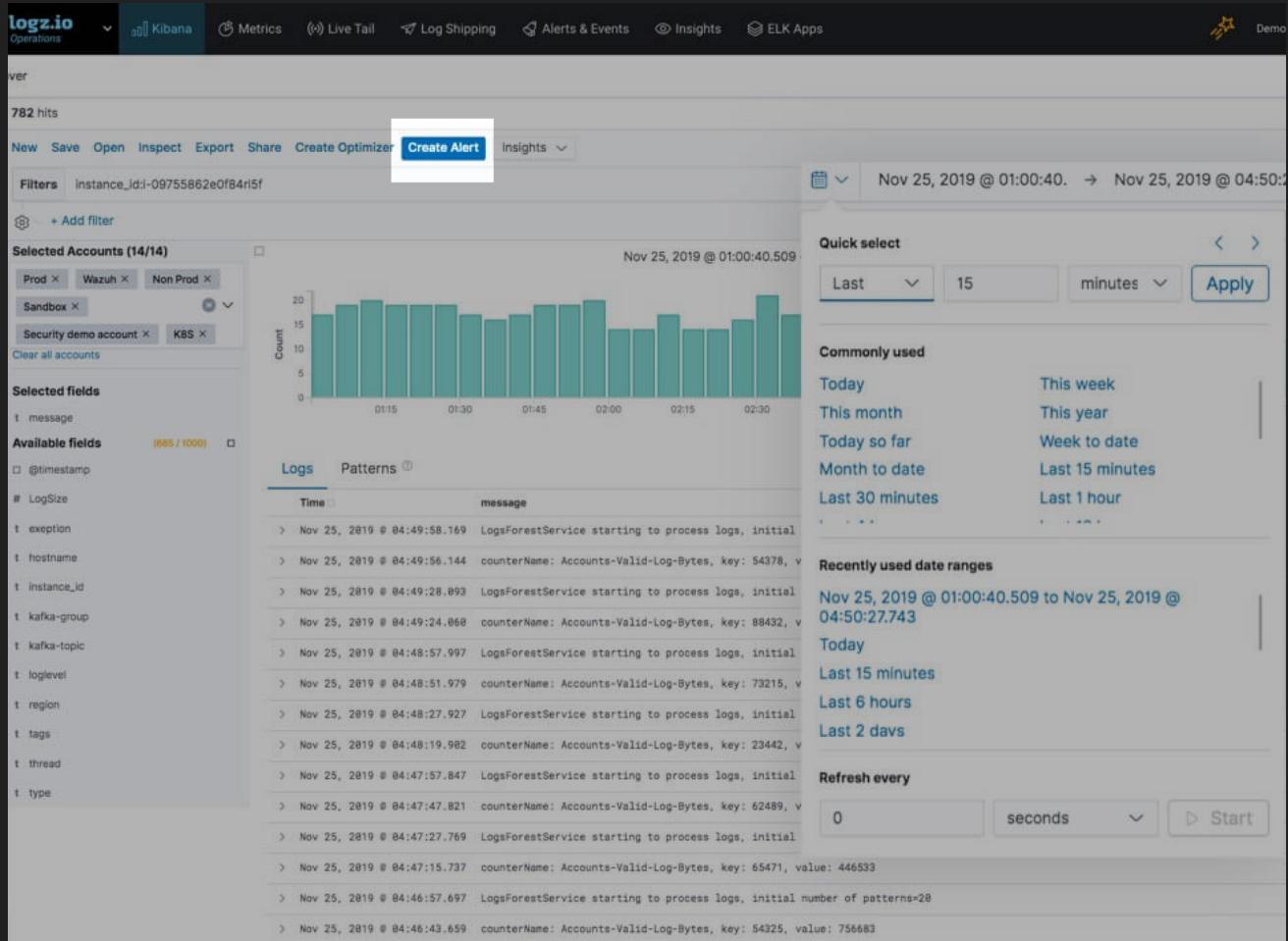


Logs por data

Queries

Alertas no slack ou email

Dashboards



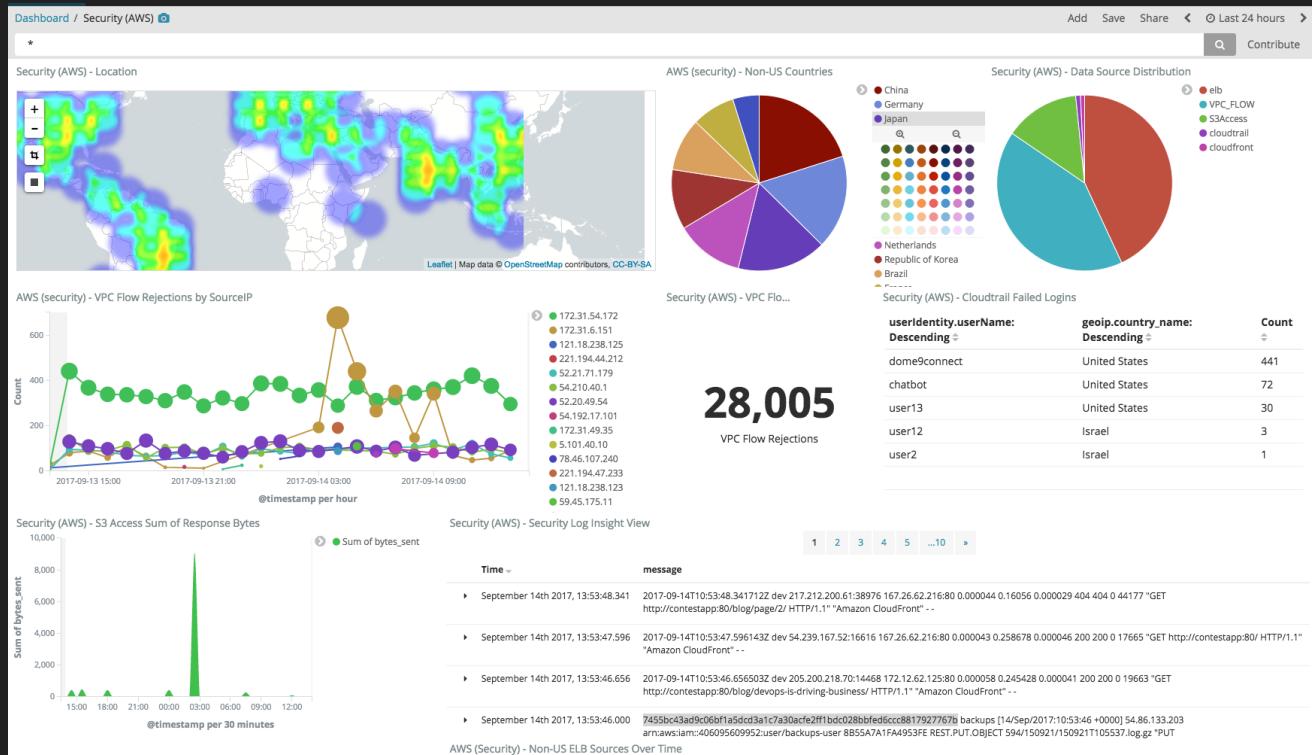


Logs por data

Queries

Alertas no slack ou email

Dashboards



To do list de nova feature

-  Mapear casos de erro
-  Criar logs no código
-  Criar alertas para os logs na ferramenta de monitoramento
-  Acompanhar os alertas

Melhorias para aplicar no seu projeto

Ferramenta de monitoramento

Protocolo Error do Swift

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

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

⚠️ Você deve mascarar informações sensíveis do usuário ⚠️

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


error.localizedDescription

```
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

Ferramenta de monitoramento

Protocolo Error do Swift

Não usar nil como erro

Retornar nulo 🤢

```
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 

```
struct 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):  
                MyLoggerWrapperClass.log(error)  
                displayEmptyState(reason: .network)  
        }  
    }  
}
```

Melhorias para aplicar no seu projeto

Ferramenta de monitoramento

Protocolo Error do Swift

Não usar nil como erro

Separar código feliz do tratamento de erro



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 monitoramento 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