



**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Đa nền tảng với Cloud Computing

Các ứng dụng muốn chinh phục hàng triệu người dùng thì chúng cần kết nối với Internet:

- liên lạc giữa các bạn bè (Facebook),

- liên lạc giữa ứng viên và công việc (LinkedIn, Outlook)

- liên lạc với người nổi tiếng (Twitter, Instagram)

Để phục vụ số lượng lớn người dùng, đem lại nội dung phong phú, cập nhật thông tin liên tục => cloud computing

Xây dựng backend cho ứng dụng cần được làm trước khi phác thảo giao diện cho ứng dụng.

Case study: lựa chọn công nghệ backend để phát triển ứng dụng đa nền tảng Domus - kết nối mọi người trong gia đình

# Thuật ngữ

Thiết bị	Điện thoại thông minh hoặc máy tính bảng
Client	Một ứng dụng chạy trên thiết bị
User (Người dùng)	Người cài đặt ứng dụng trên thiết bị
Thành viên	Một user đã đăng ký mình là thành viên của gia đình (user đã kích hoạt)
FCM	Firestore Cloud Messaging
JWT	JSON Web Token
PaaS	Platform as a Service

# Mục lục

1. Giới thiệu
2. Mô tả sản phẩm
3. Chiến lược quyết định
4. Ước lượng chi phí và giới hạn
5. Lợi nhuận

# 1. Giới thiệu

## Cross-Platform Mobile App Development



# 1. Giới thiệu

Mục đích:

giới thiệu quy trình chọn lựa các công nghệ cloud để xây dựng backend cho ứng dụng đa nền tảng

Trình bày các bước để ước lượng chi phí và lợi nhuận khi áp dụng Cloud để phát triển ứng dụng

Mục tiêu:

- hiểu được và tuân thủ GDPR (General Data Protection Regulation)
- nắm được các ưu nhược điểm của các công nghệ Cloud computing
- Ước lượng được chi phí để vận hành hệ thống cloud

# 1. Giới thiệu

## Bối cảnh

Các mạng xã hội phổ biến chỉ kết nối người dùng theo chiều ngang (cùng một sở thích, cùng thuộc địa điểm rộng lớn cỡ lục địa)

Case study: Xây dựng mạng xã hội kết nối theo chiều dọc (những thế hệ của một gia đình)

=> Domus - Một ứng dụng kết nối thành viên gia đình, cho phép:

- Nhắn tin tán gẫu (chat)
- Trao đổi danh sách mua/trả hàng
- Quản lý việc chia sẻ thẻ ghi nợ giữa các thành viên

## 2. Mô tả sản phẩm

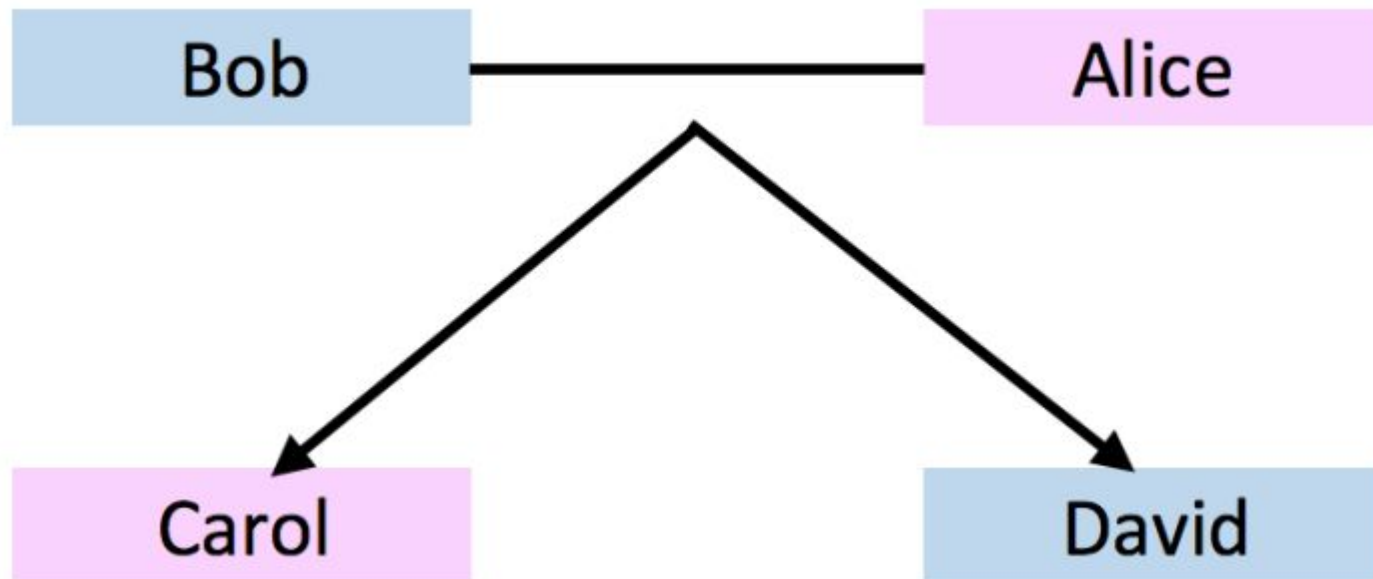




## 2. Mô tả sản phẩm

### 2.1 Kịch bản

Một gia đình điển hình có 4 thành viên: Bob, Alice và hai con Carol và David



## 2. Mô tả sản phẩm

### 2.2. Use case

Bảng sau liệt kê tất cả các use case đầy đủ của ứng dụng Domus:

Use case	Mô tả
UC1.1. Register	Người dùng đăng ký sử dụng với việc gửi thông tin: tên, email, password
UC1.2 Login	Người dùng xác nhận tài khoản qua email và sau đó có thể đăng nhập với email và mật khẩu
UC1.3 Tự đăng nhập	Sau một lần đăng nhập đầu, lần sau có thể vào thẳng ứng dụng mà không cần đăng nhập lại
UC1.4. Đổi mật khẩu	Có thể đổi mật khẩu theo ý người dùng

# 2. Mô tả sản phẩm

## 2.2. Use case

Use case	Mô tả
UC2.1. Add Family	Người dùng tạo ra một gia đình, trở thành thành viên gia đình, người tạo cũng là quản trị viên
UC2.2 Invite	Thành viên mời các người dùng khác qua email để tham gia vào gia đình của mình trên Domus
UC2.3 Join a family	Người dùng chấp nhận thư mời tham gia gia đình
UC2.4. Invite by Facebook	Có thể mời qua tài khoản Facebook
UC2.5. Quit	Rời bỏ khỏi gia đình để vào gia đình khác (một người dùng có thể là thành viên của nhiều gia đình)
UC2.6 Assign roles	Gán vai trò cho các thành viên (cha mẹ hoặc con cái hoặc ông bà)

## 2. Mô tả sản phẩm

### 2.2. Use case

Các use case khác có thể xem ở phần phụ lục (Appendix)

Use case	Mô tả
UC3.1. Send a message	Gửi tin nhắn giữa các thành viên gia đình
UC3.2 Send an image	Gửi ảnh giữa các thành viên trong gia đình
UC3.3 Save an image	Thành viên lưu ảnh vào thiết bị của mình

# 2. Mô tả sản phẩm

## 2.3 Phân tích yêu cầu

Dựa trên các use-cases trên, ta có thể thấy hệ thống (cả backend và ứng dụng) có các yêu cầu sau:

1. Quản lý chặt chẽ người dùng (UC1.x), các vai trò của người dùng không được chồng lấn nhau (UC2.3, UC2.6)
2. Lưu trữ dữ liệu có cấu trúc chặt chẽ và nhất quán vì đây là các thông tin rất hay được cập nhật bởi người dùng (tiền bạc, chi tiêu)
3. Cũng vì thông tin rất quan trọng nên cần bảo mật rất cao
4. Ứng dụng có khả năng trao đổi dữ liệu giữa các client khác nhau
5. Ứng dụng có khả năng hoạt động offline
6. Ứng dụng phải tuân thủ luật pháp sở tại và quốc tế (General Data Protection Regulation)

# 2. Mô tả sản phẩm

## 2.4 General Data Protection Regulation

GDPR là Bộ luật bảo vệ dữ liệu chung (General Data Protection Regulation) được ban hành 2018

Có hiệu lực tại Liên minh châu Âu, UK, Chile, Japan, Brazil. Mỹ cũng có bộ luật tương tự

1. Quyền được thông báo.
2. Quyền được truy cập
3. Quyền được cải chính
4. Quyền được xóa bỏ
5. Quyền được giới hạn xử lý
6. Quyền được luân chuyển dữ liệu
7. Quyền được phản đối
8. Các quyền liên quan đến việc tự ra quyết định bao gồm lược tả



### 3. Chiến lược quyết định



# 3. Chiến lược quyết định

Các chiến lược cần được quyết định trước khi xây dựng Backend:

1. Lựa chọn Client-server hay Point to Point hay Monolith hoặc Microservices
2. Lựa chọn quy trình phát triển phần mềm nào
3. Lựa chọn RESTFUL API hay SOAP
4. Lựa chọn SQL hay NoSQL
5. Lựa chọn cơ chế phù hợp cho push
6. Lựa chọn cơ chế cho offline và back online
7. Lựa chọn cơ chế cho đồng bộ dữ liệu
8. Lựa chọn cơ chế bảo mật
9. Lựa chọn cơ chế bảo trì





# 3. Chiến lược quyết định

## 3.1 Network Architecture

Có một số mô hình chính để quyết định kiến trúc cho backend: Client-Server, Point-2-Point, Microservice

Client-Server, cụ thể là PaaS (Platform as Service) được chọn vì

- Dễ dùng
- Năng lực lưu trữ gần như vô hạn
- Giảm chi phí thay vì đầu tư từ đầu
- Nhanh chóng, khả chuyển, đảm bảo khả năng mở rộng

Point to Point: mỗi thiết bị có quyền ngang hàng nhau. Nếu gia đình có số thành viên ít, đây vẫn có thể lựa chọn

Nhưng nếu tất cả thiết bị đều hỏng hoặc gỡ cài đặt => toàn bộ dữ liệu sẽ mất hết.



### 3. Chiến lược quyết định

Kiến trúc Microservice là mẫu thiết kế đang dần trở nên phổ biến trong những năm gần đây nhờ khả năng module hóa và khả năng mở rộng

- Các component có thể được phát triển, test, deploy và scale độc lập
- Một component có thể chọn stack công nghệ và ngôn ngữ lập trình của riêng nó.

Nhược điểm: công nghệ phức tạp và khó để học hơn.

- Thời gian phát triển là lâu hơn các phương án khác
- Yêu cầu cơ sở hạ tầng phức tạp. Thông thường sẽ yêu cầu nhiều container (Docker) và nhiều máy ảo để chạy.



### 3. Chiến lược quyết định

Kiến trúc một khối là mẫu thiết kế được dùng nhiều nhất trong giới lập trình web hiện nay bởi tính đơn giản của nó khi phát triển và khi deploy.

- Dễ phát triển vì các stack công nghệ thống nhất ở tất cả các layer.
- Dễ test do toàn bộ project được đóng gói trong một package nên dễ dàng chạy test integration và test end-to-end.
- Deploy đơn giản và nhanh chóng nếu chỉ có một package để bận tâm.

Nhược điểm: Khó lòng bảo trì, thời gian khởi động lâu và tốn CPU hơn, các thành phần quan trọng không thể scale riêng



# 3. Chiến lược quyết định

## 3.1 Network Architecture

Trong các mô hình Client-Server, chọn lựa PaaS

- Tiết kiệm thời gian lập trình.
- Tiết kiệm chi phí.
- Dễ dàng xây dựng phần mềm, quản lý, phân tích dữ liệu cùng lúc.

Nhược điểm:

- Không có sự quản lý, kiểm soát chuyên sâu đối với dữ liệu.
- Khó khăn khi thay đổi nhà cung cấp.
- Đáp ứng với bản cập nhật của nhà cung cấp.



# 3. Chiến lược quyết định

## 3.1 Network Architecture

Có nhiều nhà cung cấp dịch vụ PaaS, gồm:

Service	AWS	GCP	Microsoft Azure	Heroku
NoSQL storage	DynamoDB	Cloud Datastore	Cosmos DB or Stockage Table	MongoHQ
Serverless computation	Lambda	Cloud Functions*	Functions	Dynos
File storage	S3	Cloud Storage	Stockage Blob	Bucketeer
Communication	API Gateway	Google Endpoints	API Apps	WSO2 API Cloud*
Users management	Cognito	Firebase	Mobile Apps	Auth0



# 3. Chiến lược quyết định

## 3.2 Quy trình phát triển phần mềm

### Agile

- Phù hợp với các dự án có những yêu cầu không được xác định rõ ràng và có thể thay đổi thường xuyên.
- khách hàng có thể được xem trước từng phần dự án trong suốt quá trình phát triển
- Tỷ lệ thành công thường cao hơn.

### Nhược điểm:

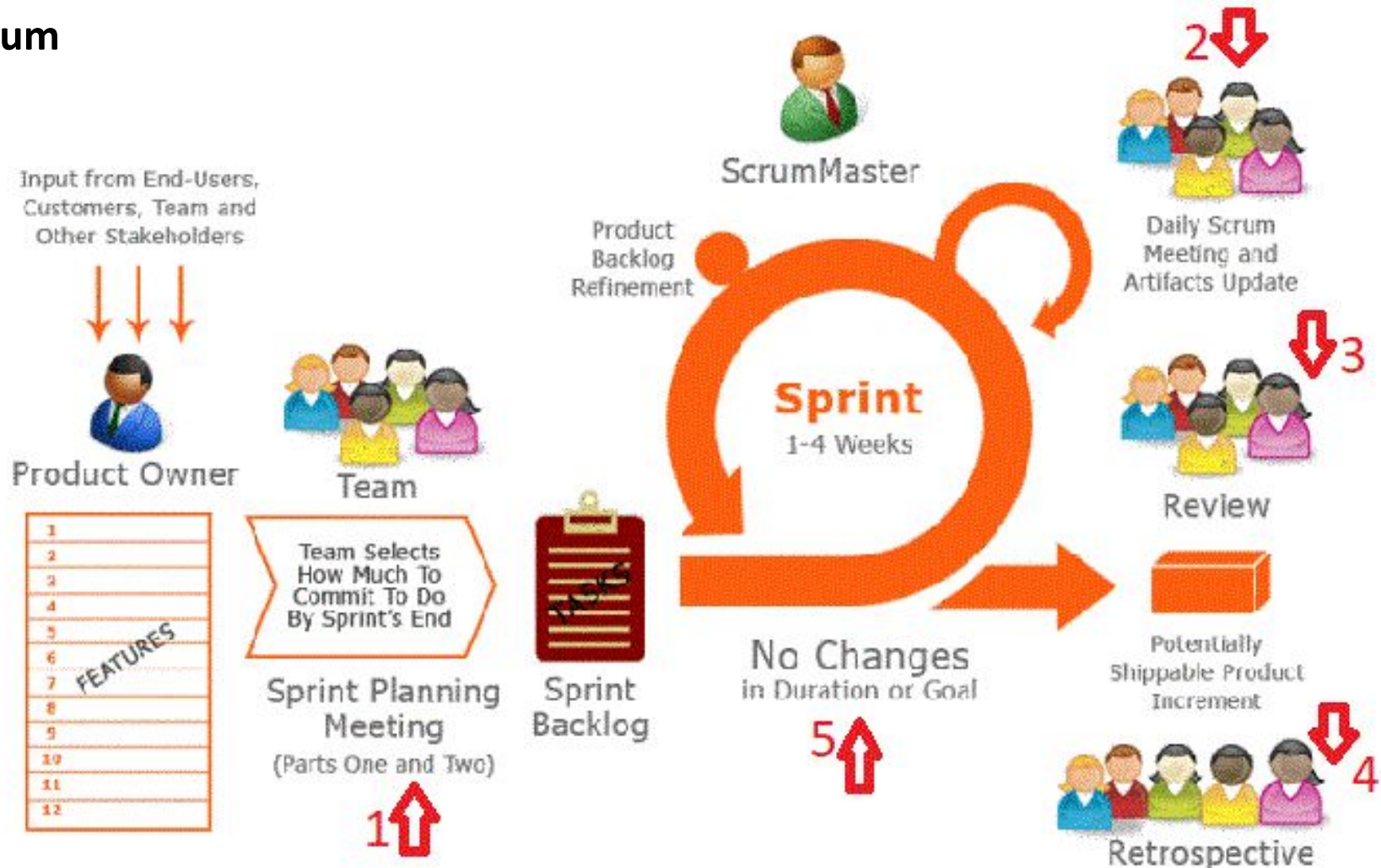
- Đòi hỏi phải có sự tham gia của những người nắm vững về Agile
- Quy mô nhân lực thường giới hạn từ 7 đến 10 người
- Số lượng yêu cầu có thể nhiều và khó quản lý nếu như nó bao gồm nhiều khía cạnh khác nhau về dự án.



# 3. Chiến lược quyết định

## 3.2 Quy trình phát triển phần mềm

### Scrum



# 3. Chiến lược quyết định

## 3.2 Quy trình phát triển phần mềm

Là một đội làm ứng dụng di động nói chung và ứng dụng đa nền tảng nói riêng thì:

- Một nhóm nhỏ dưới 10 người làm trong cùng một không gian với nhau
- Luôn giữ cơ hội trao đổi thảo luận, gặp gỡ nhau thường xuyên
- Tin tưởng giữa các thành viên
- Trao đổi thẳng thắn, chân thành, bình đẳng lẫn nhau.



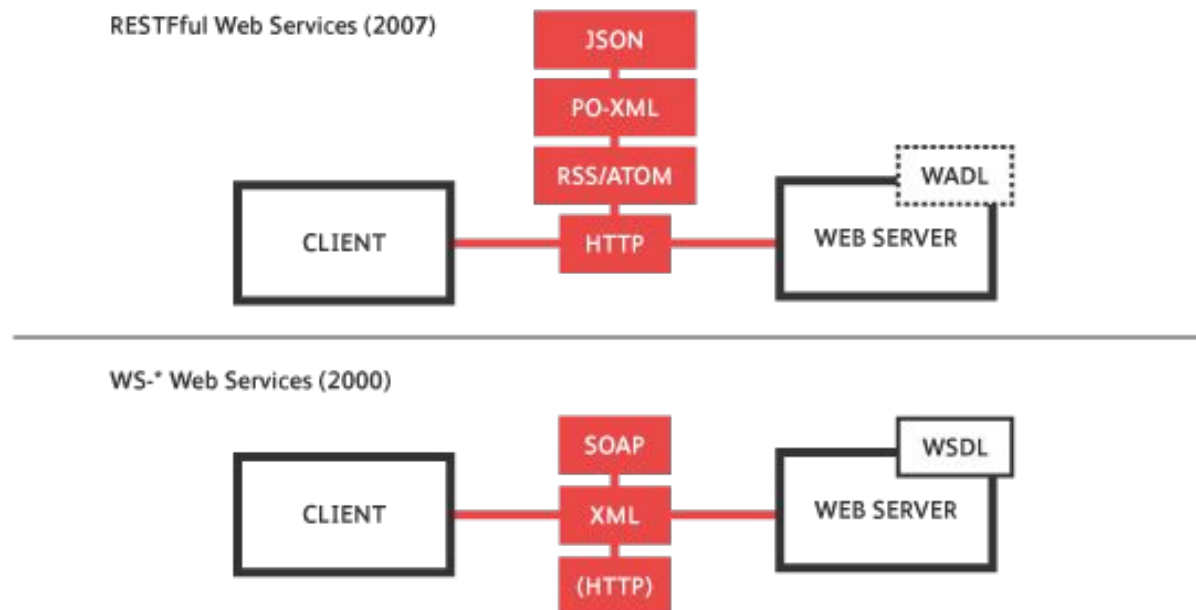


# 3. Chiến lược quyết định

## Lựa chọn giữa RESTful và SOAP Web Services

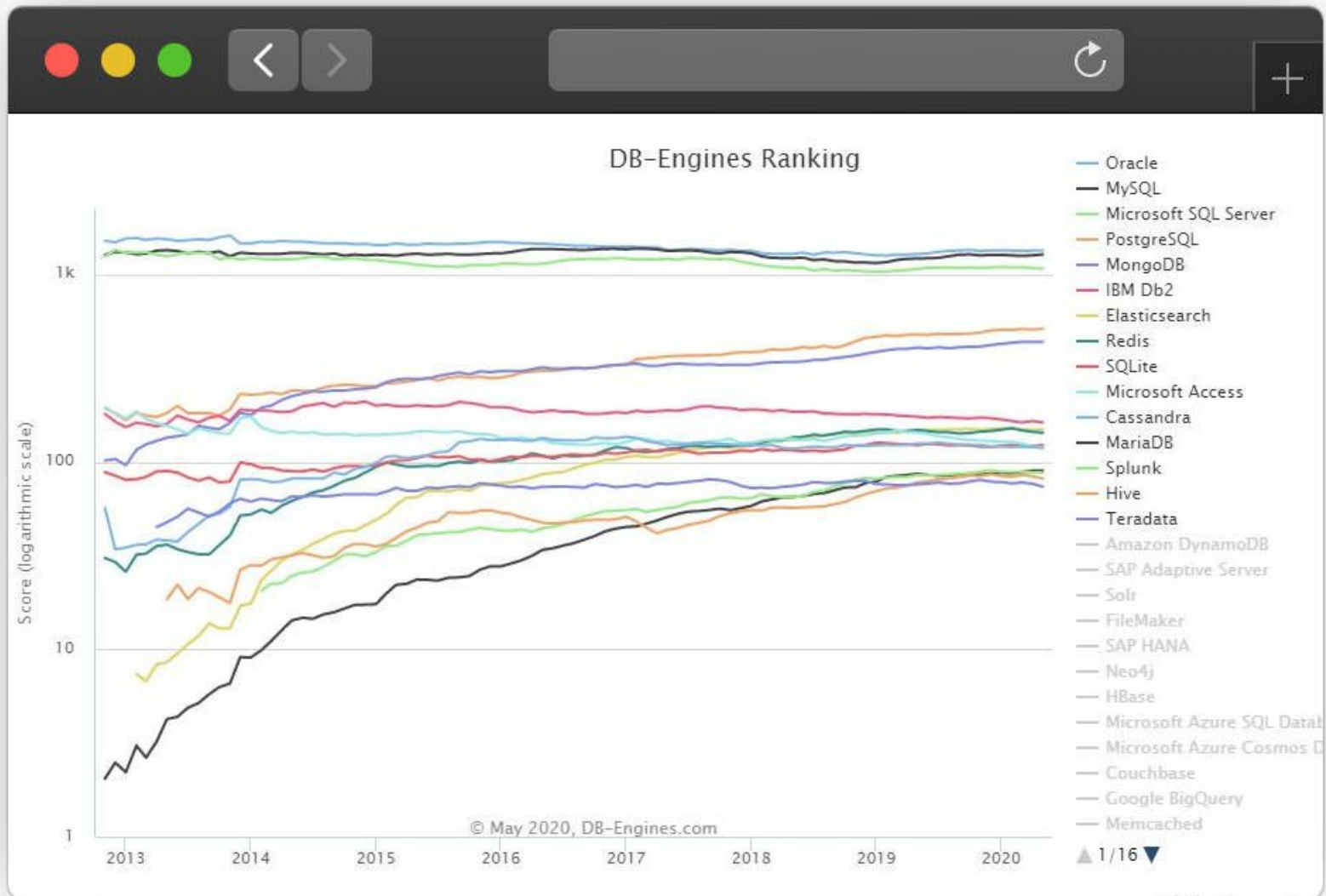
thường dùng hai giải pháp webservice chính để giao tiếp với ứng dụng web là: SOAP (Simple Object Access Protocol) và REST (Representational State Transfer).

Việc lựa chọn giải pháp nào còn tùy thuộc vào người phát triển hệ thống và từng trường hợp hệ thống cụ thể.



# 3. Chiến lược quyết định

## Lựa chọn giữa SQL và NoSQL



# 3. Chiến lược quyết định

## Lựa chọn cơ chế phù hợp cho push

Sử dụng Push notification sẽ giúp thông báo nhanh chóng hơn và giảm tải cho phía backend.

Push notification phối hợp với các thao tác người dùng để giúp hệ thống điều hướng người dùng sang các trang khác nhau.

Có thể vẫn nhận được thông báo khi người dùng không bật app

Cần chọn lựa cơ chế phân loại Push Notification:

- Có Push notification chỉ đơn giản là thông báo mà không điều hướng
- Có Push notification chỉ điều hướng khi có chấp nhận của người dùng
- Có Push notification cho phép điều hướng nhưng không quay lại được trang cũ
- Thông tin từ Push Notification không nên là các thông tin quá riêng tư hoặc nhạy cảm



# 3. Chiến lược quyết định

## Lựa chọn cơ chế cho offline và back online

Thiết bị vẫn có thể mất kết nối tại một thời điểm nào đó trong ngày

Cần phân loại những chức năng nào phải:

- hủy ngay công việc đang làm và yêu cầu client gửi lại từ đầu khi có mạng
- đợi đến khi có mạng và tiếp tục tại thời điểm dang dở
- Chấp nhận cho client xử lý offline tiếp, khi có mạng lại sẽ nhận lại dữ liệu mới.

Chẳng hạn trong ứng dụng Domus, vẫn có các chức năng được phép hoạt động offline: Add message, Add list, Add product, Edit product, Check product, Remove product, Add refund, Add card.



# 3. Chiến lược quyết định

## Lựa chọn cơ chế cho offline và back online

Một khi có mạng lại, thiết bị sẽ cần có thời gian để “hòa nhập”:

- Tránh gửi ồ ạt dữ liệu, chỉ gửi những gì quan trọng nhất và mới nhất
- Những thông báo cũ xuất hiện có thể được bỏ qua nếu cần
- Cần chấp nhận việc bị mất mát dữ liệu hoặc bị bỏ lỡ

Nhìn chung ở các hệ thống chia sẻ dữ liệu, ta chỉ đạt được tối đa hai trong số các tiêu chí sau:

- Nhất quán (**C**onsistency)
- Hiện hữu (**A**vailability)
- Truy cập được khi offline (Tolerance to network **P**artition)

Nguyên lý CAP được Brewer đề xuất



# 3. Chiến lược quyết định

## Lựa chọn cơ chế cho đồng bộ dữ liệu

Trong trang cá nhân, thành viên Bob thực hiện việc thay đổi ảnh avatar.

Bob vào trang chat để trao đổi với Alice. => Ảnh cá nhân của Bob trong danh sách các tin nhắn có được đổi không và cách đổi như thế nào?

**Tình huống 1:** Server nhận ảnh avatar mới và chỉ cập nhật ảnh, không cập nhật link

Khi vào cửa sổ chat, ảnh avatar của Bob nếu vẫn truy cập link cũ sẽ vẫn thấy ảnh cũ (cache dữ liệu). => Cách gì để giải quyết?

**Tình huống 2:** Server nhận ảnh avatar mới và cập nhật đổi cả link ảnh.

Khi vào cửa sổ chat, làm thế nào để client biết rằng phải đổi link truy cập ảnh?

↓ **Tình huống 3:** Alice vào cửa sổ chat, làm thế nào client ở Alice biết Bob đã đổi ảnh?

# 3. Chiến lược quyết định

## Lựa chọn cơ chế bảo mật

Bảo mật là vấn đề quan trọng của mọi ứng dụng có đông người dùng. Vì khách hàng càng đông thì càng có nguy cơ bị tấn công

Bên cạnh đó việc dữ liệu càng nhiều thì client càng phải lưu trữ nhiều thông tin để giảm tải cho server.

=> Việc bảo mật cần phải làm ở cả phía server và client.

a) Dữ liệu để đăng nhập

b) JSON Web Token: khi nhận đủ thông tin, server sẽ trả về các JWT và client sẽ lưu trữ các thông tin này để sử dụng.

c) Đăng xuất: một khi đăng xuất sẽ xóa hết các thông tin quan trọng.

d) Chống cùng truy cập: một số hệ thống ngăn cấm một tài khoản đăng nhập trên hai máy khác nhau.



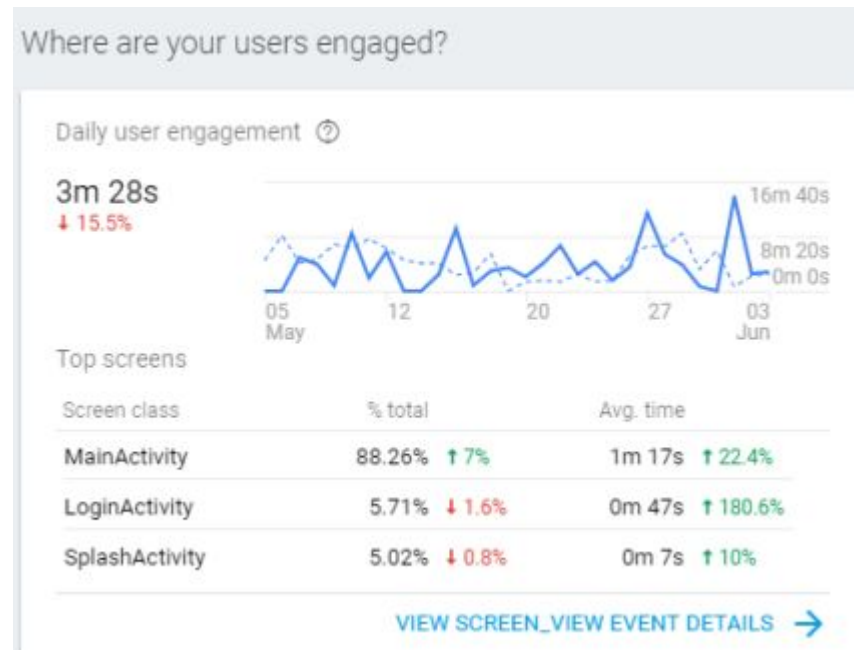
# 3. Chiến lược quyết định

## Lựa chọn cơ chế hỗ trợ bảo trì/nâng cấp

Một khi sản phẩm được tung ra thị trường thì ta cần thu thập đủ thông tin để phục vụ việc bảo trì/nâng cấp:

a) Google Analytics: thu thập thời gian người dùng dành cho các màn hình

Đã có các công cụ hỗ trợ cho cả RN và Flutter





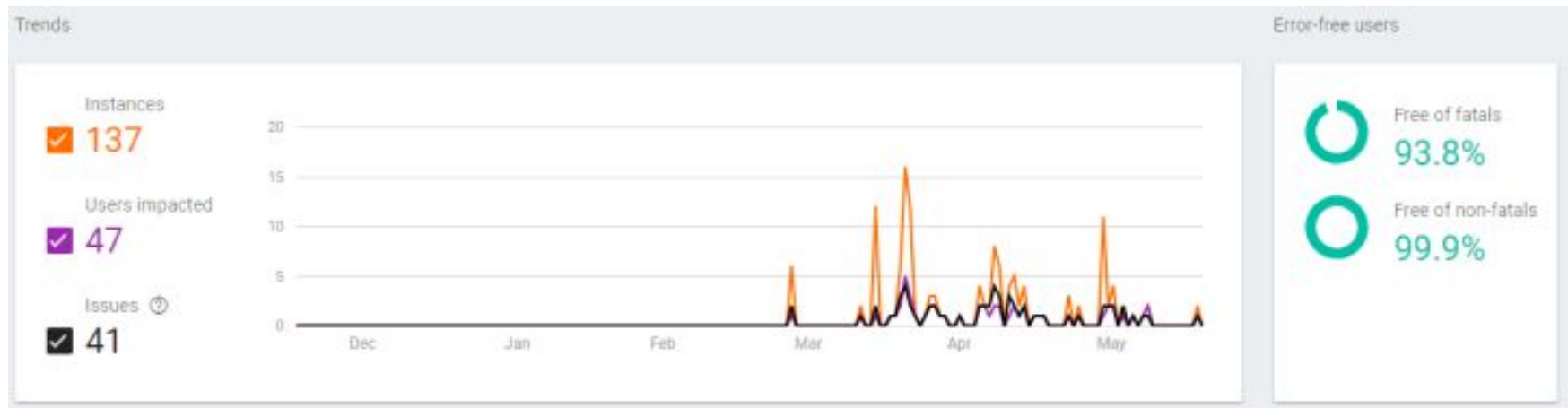
# 3. Chiến lược quyết định

## Lựa chọn cơ chế hỗ trợ bảo trì/nâng cấp

b) Crash reporting: thu thập dữ liệu của ngoại lệ khi ứng dụng bị thoát đột ngột (Crash).

Dữ liệu đó sẽ gửi về ngay cho đội phát triển một khi client kết nối được với mạng ở lần gần nhất

Đã có các công cụ hỗ trợ cho cả RN và Flutter

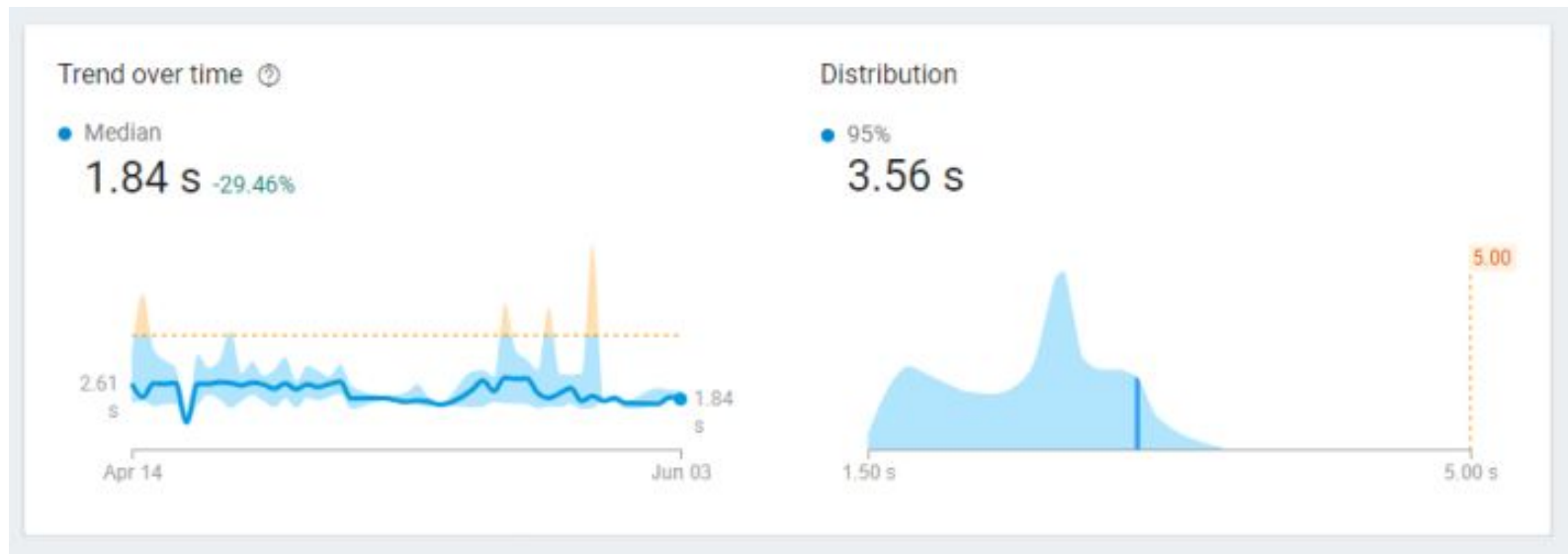


# 3. Chiến lược quyết định

## Lựa chọn cơ chế hỗ trợ bảo trì/nâng cấp

c) Phân tích hiệu năng: đo đặc thời gian bất ứng dụng, thời gian đợi phản hồi HTTPS request.

Đã có các công cụ hỗ trợ cho cả RN và Flutter

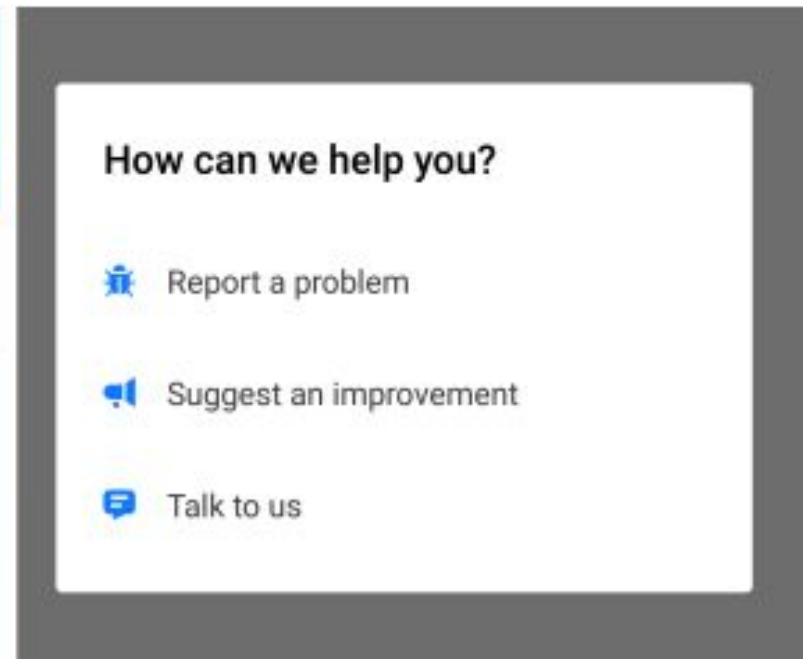
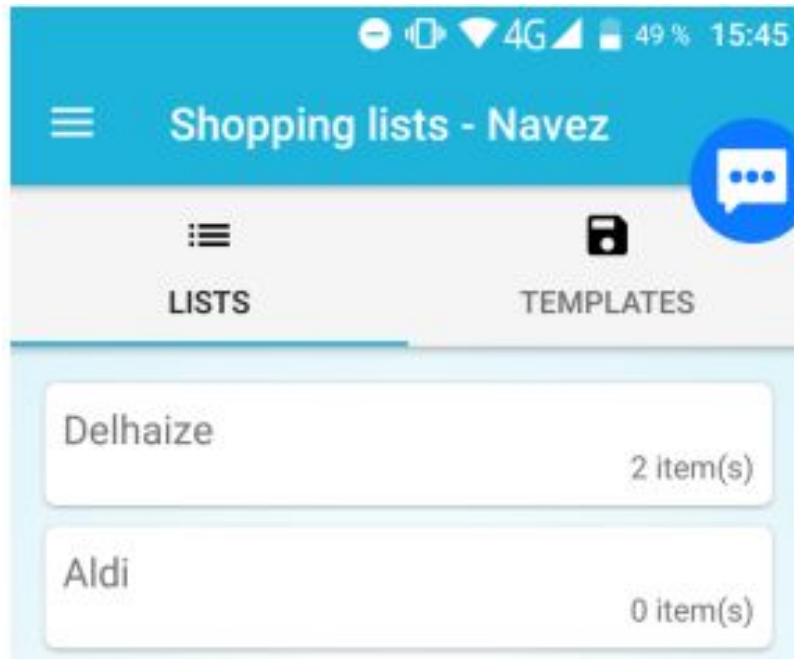


# 3. Chiến lược quyết định

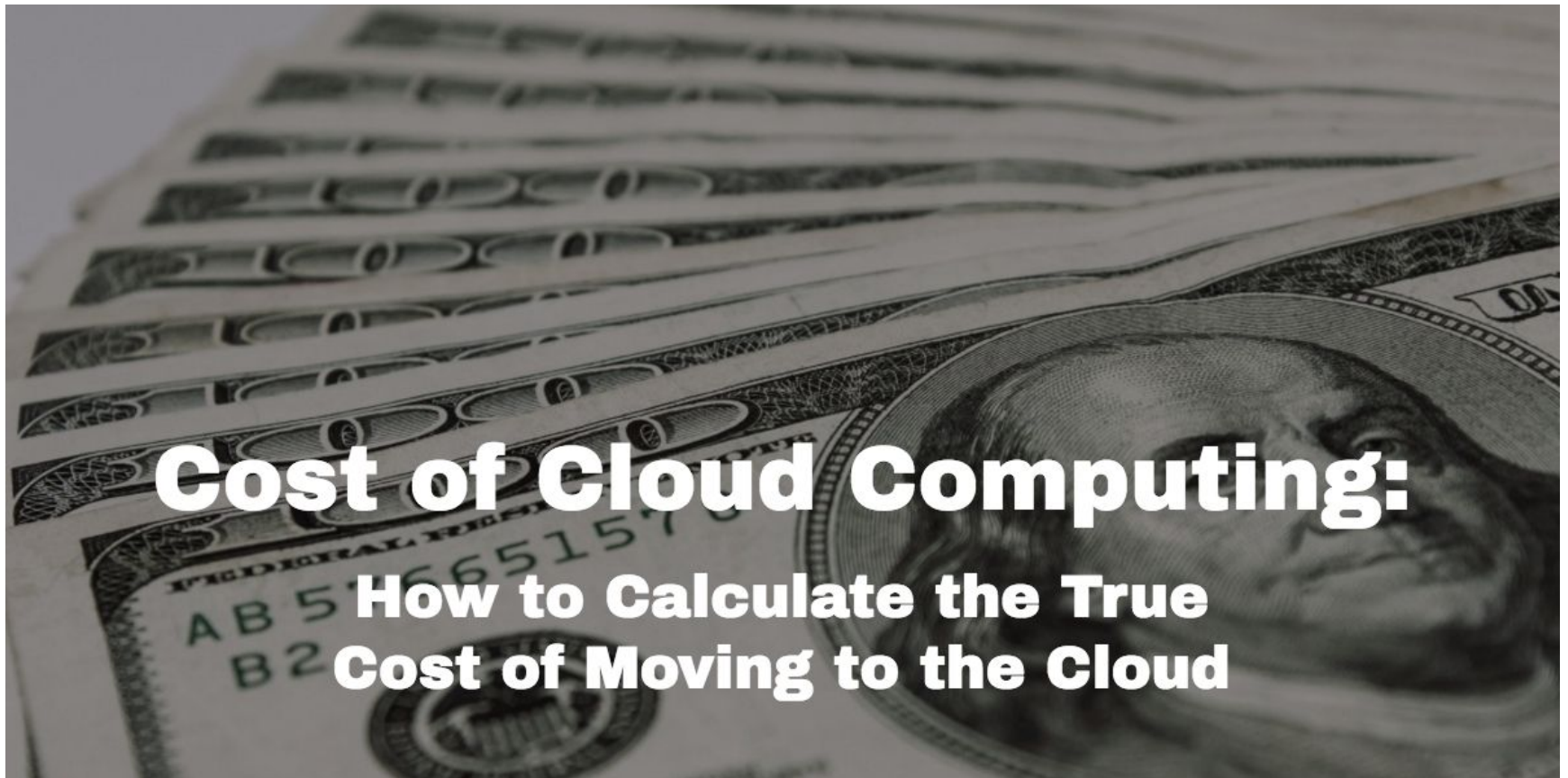
## Lựa chọn cơ chế hỗ trợ bảo trì/nâng cấp

d) Instabug: công cụ hỗ trợ thu thập ý kiến của người dùng khi có: nhu cầu đề xuất, báo cáo các lỗi, trả lời khảo sát thậm chí chat.

Đã có các công cụ hỗ trợ cho cả RN và Flutter



## 4. Ước lượng chi phí và giới hạn



Feature	Description	Approx time	Approx cost with back-end, \$ (based on \$40/h)
Login	<ul style="list-style-type: none"> <li>• Login with email</li> <li>• Login with social media</li> <li>• Forgot password option</li> <li>• Log out</li> </ul>	28 - 42 hours	1,120-1,680
File uploading	<ul style="list-style-type: none"> <li>• Upload photo</li> <li>• Upload video</li> <li>• Video playback</li> <li>• Photo view</li> </ul>	20 - 30 hours	800-1,200
Profile completion	<ul style="list-style-type: none"> <li>• Set region</li> <li>• Add info</li> <li>• Add photo</li> </ul>	23 - 29 hours	920-1,160
Profile editings	<ul style="list-style-type: none"> <li>• Edit profile</li> <li>• Change password</li> <li>• Change email</li> <li>• Add/remove credit card</li> </ul>	47 - 62 hours	1,880-2,480



Search	<ul style="list-style-type: none"> <li>• Basic search with suggestions</li> </ul>	13 - 18 hours	520-729
Basic messaging	<ul style="list-style-type: none"> <li>• Conversation details</li> <li>• Online/offline status</li> <li>• Typing status</li> <li>• Read/sent status</li> <li>• Send media files and documents</li> </ul>	160 - 170 hours	6,400-6,800
Push notifications	<ul style="list-style-type: none"> <li>• Users can receive push notifications</li> </ul>	25 - 32 hours	1,000-1,280
Basic admin panel: User management	<ul style="list-style-type: none"> <li>• See list of users</li> <li>• Edit user</li> <li>• Delete/block user</li> <li>• Create user</li> </ul>	66 - 90 hours	2,640-3,600
Basic admin panel: Payment management	<ul style="list-style-type: none"> <li>• See payments</li> <li>• Refund payments</li> </ul>	23 - 44 hours	920-1,760
Basic admin panel: Push notifications	<ul style="list-style-type: none"> <li>• Send custom push notifications</li> </ul>	8 - 14 hours	320-560



## Mobile app cost estimate: Breakdown by complex features

Feature	Description	Approx time	Approx cost with back-end (based on \$40/h)
Map	<ul style="list-style-type: none"><li>• Detect user's location</li><li>• Search on the map</li><li>• Set pickup point on the map</li></ul>	75 - 111 hours	3,000-4,440
Payments	<ul style="list-style-type: none"><li>• See balance</li><li>• List of transactions</li><li>• Add card Add PayPal</li></ul>	60 - 78 hours	2,400-3,120
Streaming	<ul style="list-style-type: none"><li>• Start/stop broadcast</li><li>• View broadcast</li><li>• Switch between broadcasts</li></ul>	90 - 140 hours	3,600-5,600
Calls	<ul style="list-style-type: none"><li>• Audio calls</li><li>• Video calls</li><li>• List of contacts</li></ul>	257 - 365 hours	10,280-14,600

## 4. Ước lượng chi phí và giới hạn

1) Phí tồn trên số người dùng active hàng tháng (MAU)

Một người active (tùy quan điểm của hệ thống backend) sẽ là người truy cập ít nhất một lần trong tháng.

Thông thường, giá của các active user ban đầu là miễn phí nếu số lượng thấp hơn ngưỡng

Giá này sẽ giảm nếu càng nhiều người dùng active.

Pricing Tier (MAUs)	Price per MAU
First 50,000	Free
Next 50,000	\$0.00550
Next 900,000	\$0.00460
Next 9,000,000	\$0.00325
Greater than 10,000,000	\$0.00250



## 4. Ước lượng chi phí và giới hạn

2) Phí tổn trên số lời gọi API

Nếu càng nhiều lần gọi đến API thì chi phí càng tăng

Cũng như dữ liệu outgoing càng lớn thì càng mất nhiều tiền (ingoing miễn phí)

Đơn giá có thể đến \$3.5 cho một triệu lần gọi

Pricing Tier (TB)	Price per GB
First 10	\$0.09
Next 40	\$0.085
Next 100	\$0.07
Next 350	\$0.05

# 4. Ước lượng chi phí và giới hạn

## 3) Phí sử dụng CSDL

$$WCU = \frac{w}{t} * [sw]$$

$$RCU = \frac{\frac{r}{t} * \left[ \left( \frac{sr}{4} \right) \right]}{2}$$

$w$  = number of item writes,

$t$  = time in second,

$sw$  = size of an item in 1KB blocks.

$r$  = number of item reads,

$t$  = time in second,

$sr$  = size of an item in 1KB blocks.

W/R CU (Write/Read Capacity Unit)

Phí trên dung lượng lưu trữ (ví dụ \$0.25/GB và miễn phí 25GB đầu)

Một số hệ thống còn tính giá thành thuê theo số lượng dữ liệu yêu cầu của tháng trước.

Chẳng hạn giá thành phải trả để phục vụ cho 10 vạn khách hàng vào tháng đầu sẽ ít hơn so với tháng thứ 12 và tháng thứ 24 (nếu vẫn còn số lượng khách đó)

# 4. Ước lượng chi phí và giới hạn

## 4) Giới hạn

Một số hệ thống giới hạn 10,000 request/s trong một đất nước/khu vực

Một số hệ thống chỉ cho phép chạy đồng thời khoảng 1,000 request tại một thời điểm trong một đất nước/khu vực





Một số hệ thống CSDL chỉ phục vụ được trong một đất nước/khu vực tối đa đến 20,000 RCU và 20,000 WCU

Dường như iOS đang tự giới hạn, nếu thiết bị nhận nhiều hơn 1 thông báo trong một giây, và ngưỡng này bị vượt trong 10-20s liên tục thì iOS tự chặn trong 1-2 giờ.

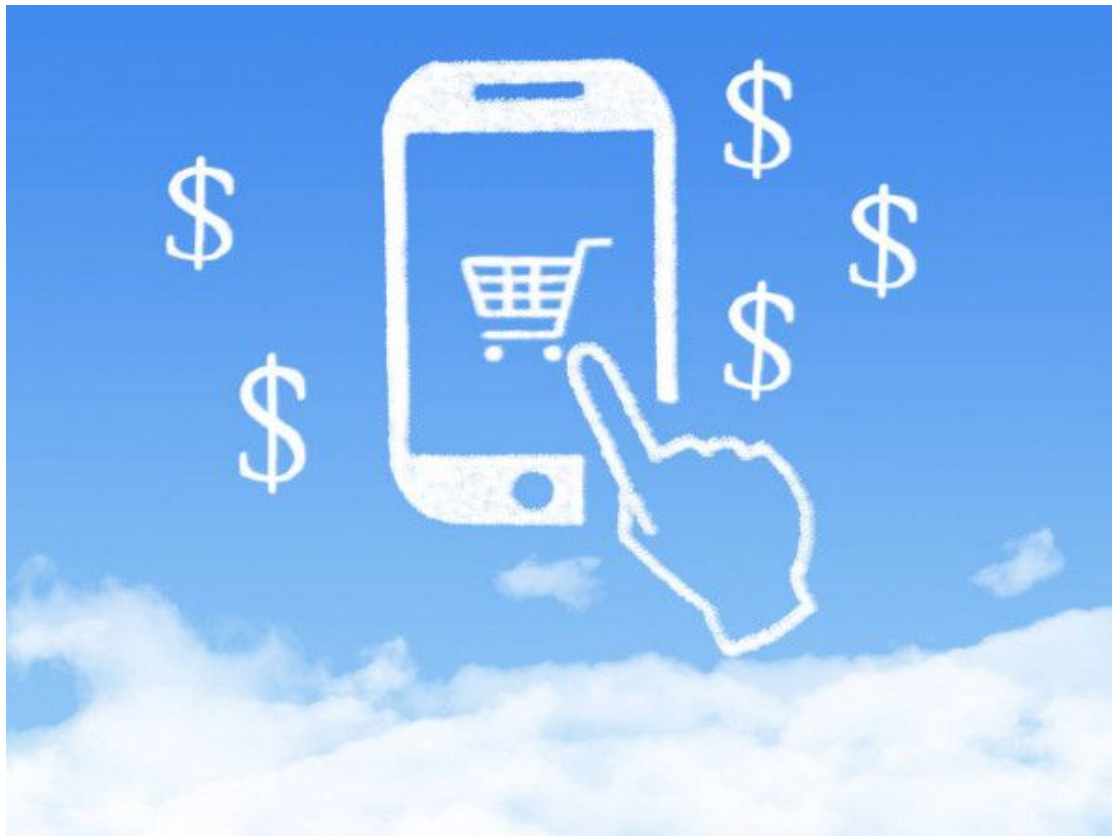
## 4. Ước lượng chi phí và giới hạn

Dựa trên một số giả thuyết, căn cứ vào giá thành phải trả, có thể dự đoán được chi phí và số người dùng trong từng tháng

Simulation		1st month	12th month	24th month	Total for 2 years
Null	\$ / 	\$1.91 100	\$2.72 100	\$3.68 100	\$66.40
Pessimistic	\$ / 	\$1.97 105	\$4.44 180	\$9.53 282	\$120.56
Neutral	\$ / 	\$2.03 110	\$7.43 314	\$21.35 656	\$222.48
Optimistic	\$ / 	\$2.16 120	\$20.09 892	\$85.87 2675	\$728.98

Simulation		1st month	12th month	24th month
Null	\$ / 	.0191	.0272	.0368
Pessimistic	\$ / 	.0188	.0247	.0338
Neutral	\$ / 	.0185	.0237	.0325
Optimistic	\$ / 	.0180	.0225	.0321

## 5. Lợi nhuận



## 5. Lợi nhuận

- Một khi chúng ta đã có nhiều người dùng, hãy tìm cách kiếm tiền từ họ.
- Lợi nhuận đó sẽ giúp chúng ta có chi phí duy trì hoạt động trong khi vẫn cung cấp các game/app miễn phí cho người dùng.
- Hãy luôn tâm niệm rằng làm ra các ứng dụng cho khách hàng là một trong các cách “an toàn nhất” để kiếm tiền (thậm chí so với cả làm ra các trò chơi)
  - Tuy vậy: đừng chọn những nước có chủ nghĩa dân tộc cực đoan.



## 5. Lợi nhuận

Có hai cách để thu lời từ ứng dụng

### **Ứng dụng trả phí**

- Thuê bao/người dùng trả trước để tải xuống và sử dụng ứng dụng
- Chìa khoá chính là nhà phát triển phải chọn giá phù hợp/giá tốt
- Việc cập nhật ứng dụng đặc biệt này phải miễn phí

### **Thử thách**

- Tạo đủ lượt tải xuống để sinh lời vì người dùng sẽ cực kỳ cẩn trọng với ứng dụng loại này



# 5. Lợi nhuận

## •Ứng dụng miễn phí

- Người dùng có thể tải miễn phí ứng dụng
  - Với Android thì sẵn sàng trả tiền cho người bán thiết bị để họ cài đặt luôn ứng dụng khi bán cho khách mua điện thoại
- Người dùng cần trả tiền cho các tính năng mở rộng
- Hoặc người dùng cần phải xem/nhấp chuột vào các quảng cáo

## •Thử thách

- Càng nhiều lượt tải của người dùng càng tốt
- Chọn lựa nhiều nhà quảng cáo khác nhau, hàng đầu vẫn là Admod/AppleAds (**TẠI SAO VẪN DÙNG CÁC NHÀ QUẢNG CÁO KHÁC**)





# 5. Lợi nhuận

## • Thử thách của ứng dụng miễn phí

- Quảng cáo sẽ che mất khung hình của trò chơi/ứng dụng vốn đã bé
- Quảng cáo có khả năng chiếm băng thông của ứng dụng vốn có nhu cầu kết nối Internet
- Mặc dù bị cấm, nhưng chúng ta cần tìm ra các chiến lược khuyến khích người dùng nhấn vào quảng cáo
  - Một khi người dùng click quảng cáo, ta thống kê và “ngầm” tặng các khuyến mãi (theo xác suất)
- Bản thân chúng ta cũng có những quảng cáo của riêng chúng ta: các ứng dụng/trò chơi mới/các thông báo khuyến mãi



# 5. Lợi nhuận

## ▪Miễn phí để chơi (với trò chơi)

- Cho phép chia sẻ lên MXH các chiến tích mà họ giành được khi chơi
- Kết hợp cơ chế trò chơi khuyến khích người chơi tiêu tiền
- Cho phép người chơi thu thập các thành tích và đổi thành các tính năng nâng cao (có giới hạn thời gian)
  - Các tính năng nâng cao đó vẫn có thể dùng tiền mua được

## ▪Khó khăn

- Rồi sẽ một ngày người chơi cảm thấy chán và rời bỏ đi.



# 5. Lợi nhuận

- Một ứng dụng \$1,99 mang đến cho nhà phát triển khoảng \$1,40 cho mỗi lượt tải xuống
- Ứng dụng cần 8.407 lượt tải xuống mỗi năm để một lập trình viên (giả sử họ một mình tự làm ứng dụng) đạt ngưỡng nghèo của Hoa Kỳ năm 2015 (11.770 đô la mỗi năm)
- Chưa tính đến các yếu tố:
  - Các nhà phát triển khác đang cạnh tranh
  - Thời gian để học công cụ và ngôn ngữ
  - Trang thiết bị khác (đồ họa, âm thanh, điện nước, điều hòa...)



# 5. Lợi nhuận

- Tập trung phát triển vào nền tảng nào trước?
- Hướng đến đối tượng nào (giới tính, lứa tuổi, sở thích, quốc tịch...)
- Giá cả của các vật phẩm (tính năng nâng cao) trong game/app
- Phương án thanh toán:
  - In-app purchase
  - Web charging
  - SMS VAT (tin nhắn giá trị gia tăng)
  - Kênh thanh toán VISA, chuyển khoản
  - Nạp thẻ cào...
  - Mobile money

# 5. Lợi nhuận

## HÃY THẢO LUẬN VỀ CÁC ƯU NHƯỢC ĐIỂM CỦA CÁC PHƯƠNG ÁN THANH TOÁN?

- In-app purchase
- Web charging
- SMS VAT (tin nhắn giá trị gia tăng)
- Kênh thanh toán VISA, chuyển khoản
- Nạp thẻ cào...
- Mobile money
- Tiền kỹ thuật số

# 5. Lợi nhuận

▪ Một vài suy nghĩ:

- Trên các ứng dụng người dùng iOS chi tiêu nhiều hơn khoảng 3 lần so với người dùng Android
- Android có thị phần lớn hơn trên phạm vi toàn cầu.
- Android được các doanh nghiệp (ở các nước đang phát triển) lựa chọn nhiều hơn
- Ở các nước phát triển như Mỹ, LTV ưa thích iOS hơn
  - Trừ thể thao và truyện tranh, nhìn chung những gì chinh phục được giới trẻ Mỹ sẽ chinh phục được giới trẻ thế giới.



Thank You  
For Your Attention



**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Phụ lục (Appendix)



# Phụ lục (Appendix)

## Các use cases còn lại của Ứng dụng Domus

UC	Description
UC 4.1	Create a shopping list
UC 4.2	Add a product to a list
UC 4.3	Check/uncheck a product
UC 4.4	Select a loyalty card at the checkout
UC 4.5	Save a list as a template at the checkout
UC 4.6	Request a refund at the checkout
UC 4.7	Keep unchecked products in a remaining list
UC 4.8	Rename a list
UC 4.9	Remove a list
UC 4.10	Switch a list to active or template
UC 4.11	Edit a product
UC 4.12	Remove a product

UC	Description
UC 5.1	Add a loyalty card
UC 5.2	Show a loyalty card
UC 5.3	Rename a loyalty card
UC 5.4	Change the logo of a loyalty card
UC 5.5	Remove a loyalty card

UC	Description
UC 6.1	Add a refund request
UC 6.2	Edit a refund request
UC 6.3	Refund a request
UC 6.4	Remove a refund request

UC	Description
UC 7.1	Log out
UC 7.2	Change profile picture
UC 7.3	Change currency