

Front-End APIs

Powering Fast-Paced Product Iterations



Speakers



Aditya Modi

Staff Software Engineer



Karthik Ramgopal

Sr Staff Software Engineer

Overview

History and evolution of frontend APIs at LinkedIn

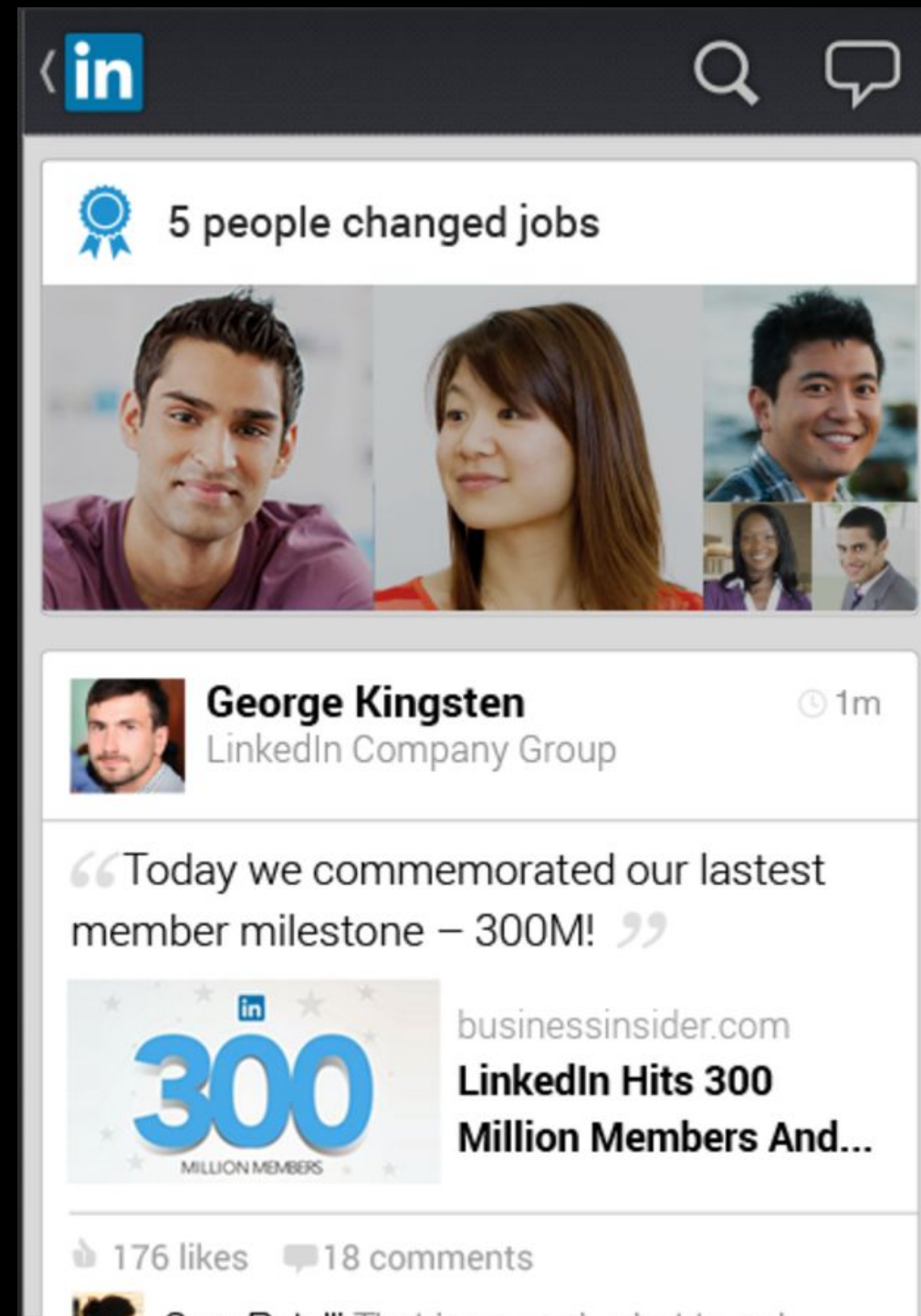
Our API structure today

Learnings and results

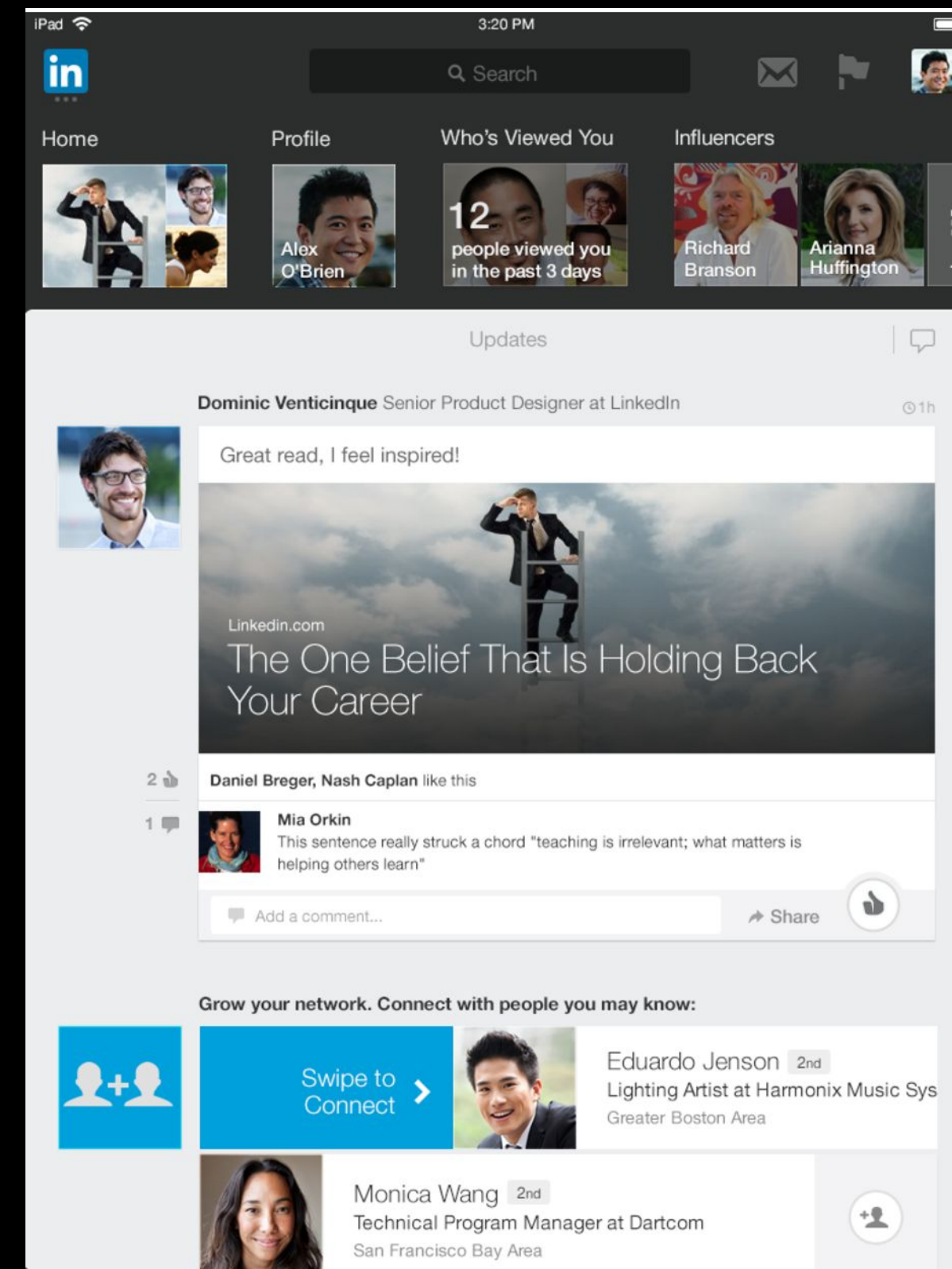
Sneak peek at the future

2 Years Ago

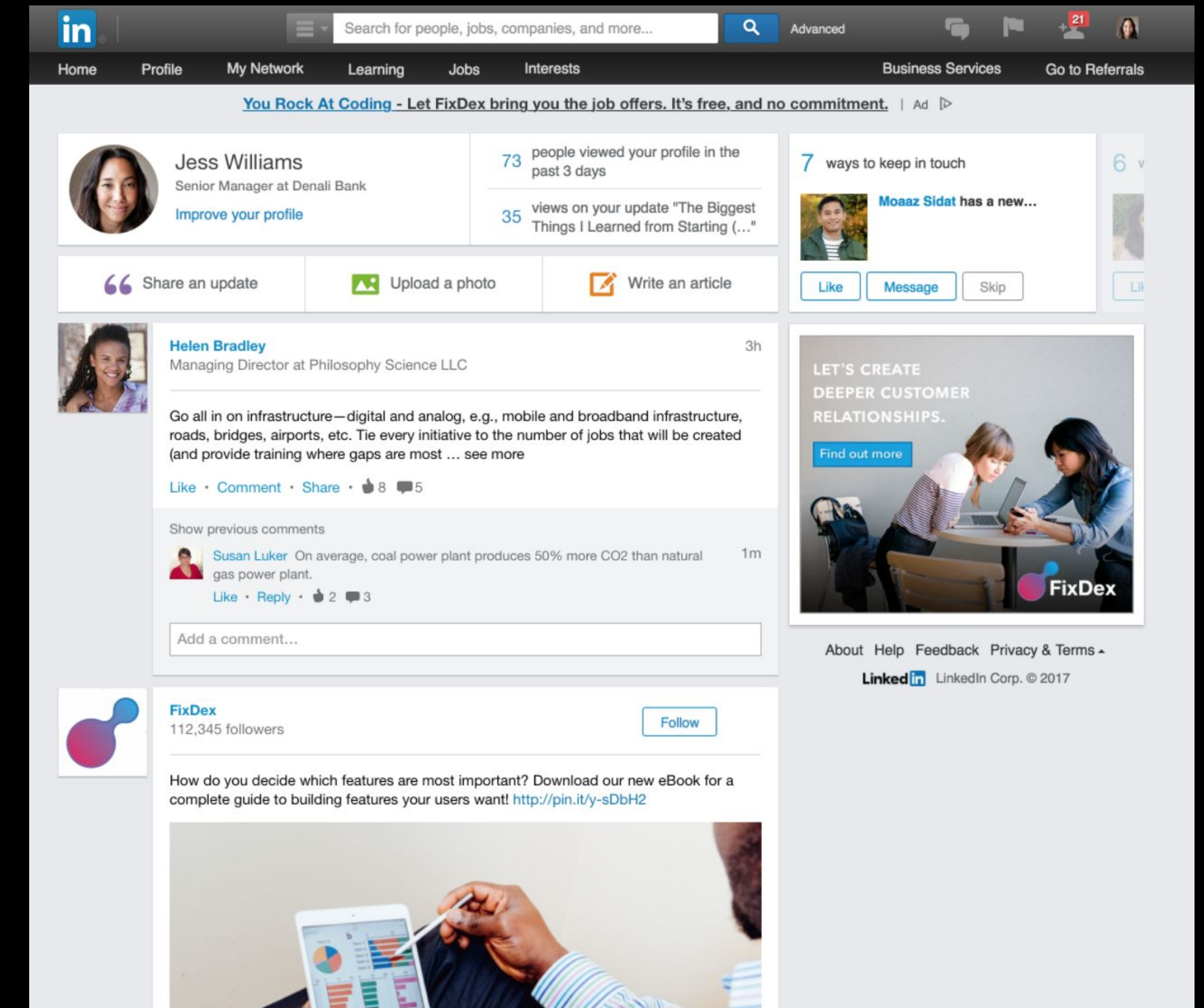
Mobile v/s Desktop



Feed on mobile

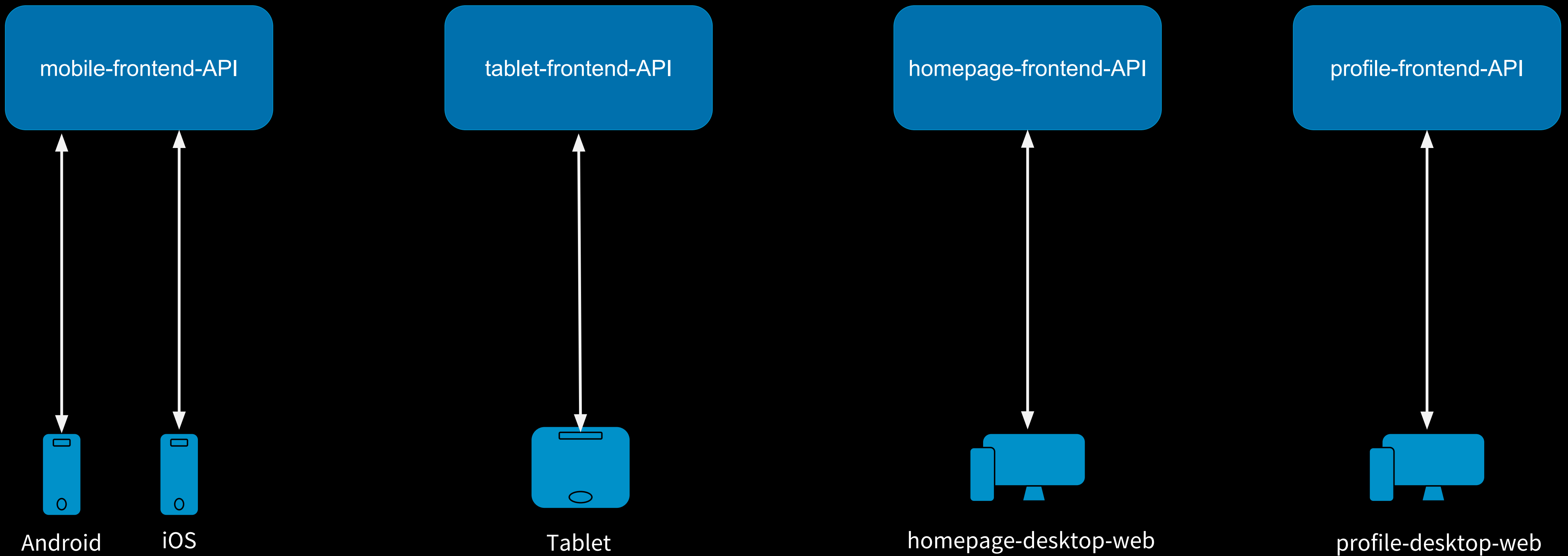


Feed on iPad



Feed on desktop

Client - Server setup

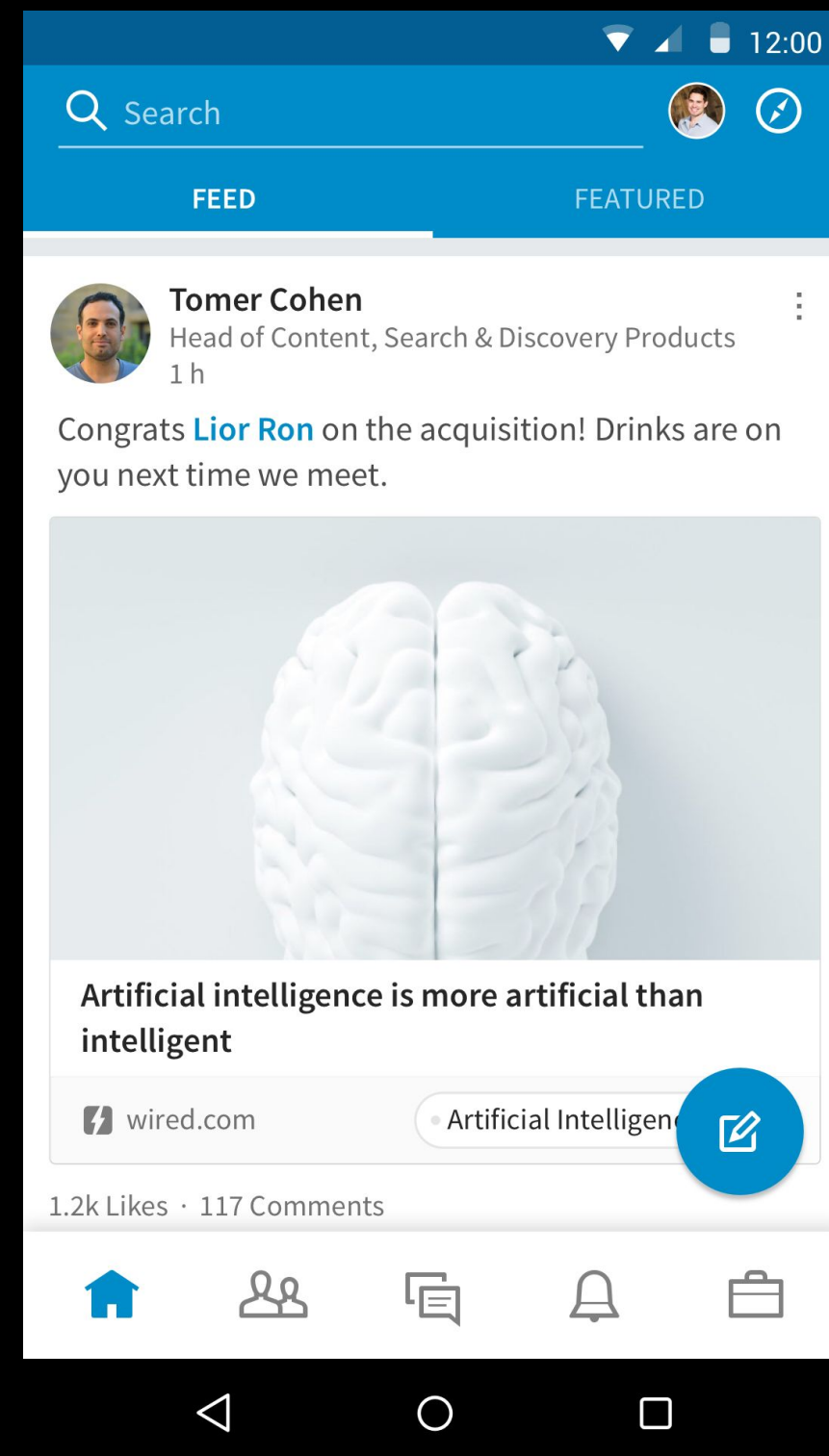


Problems?

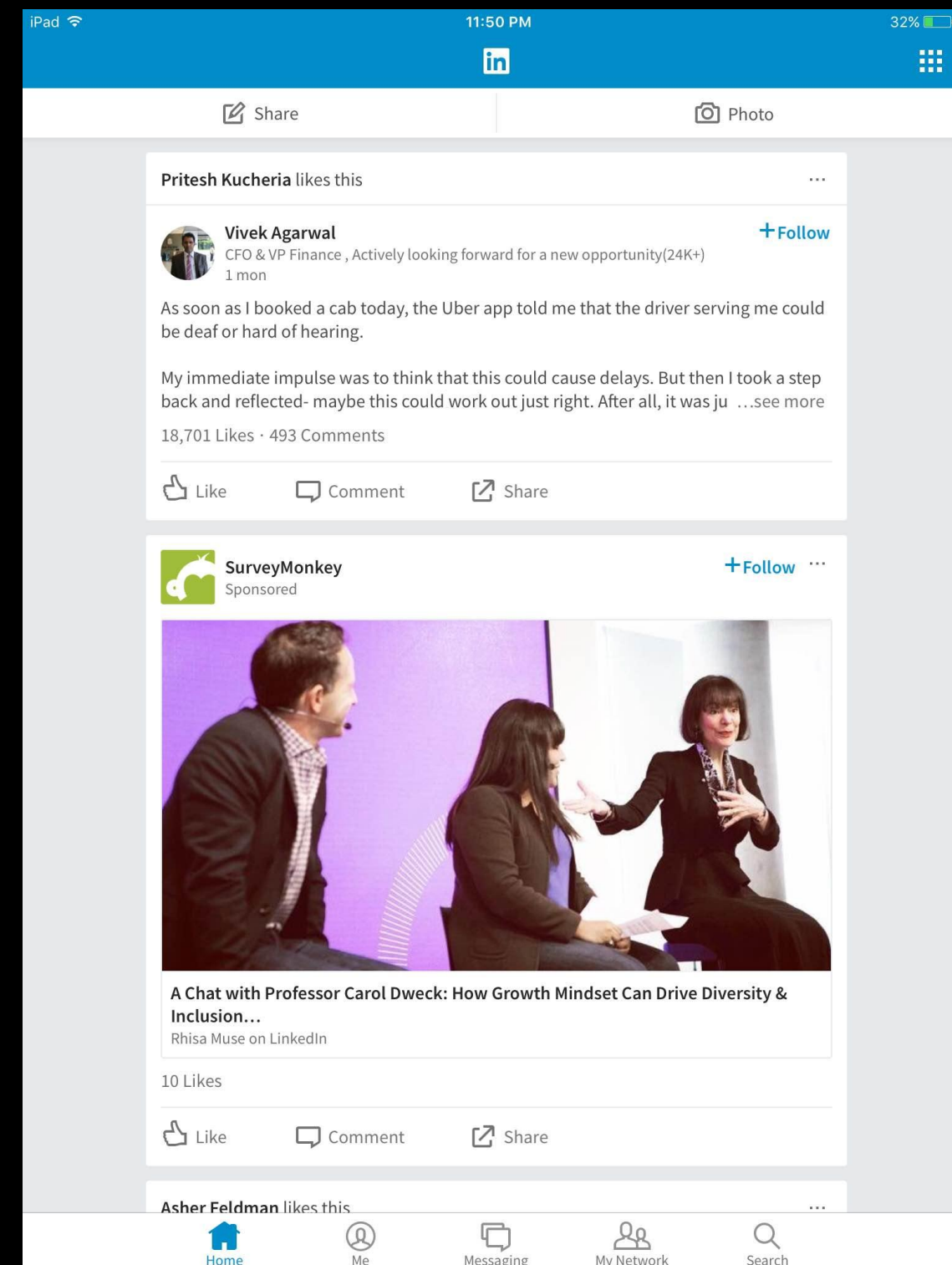
- Huge API surface and diversity
- No IDL/schema backing API
- Slow iteration speed

Today

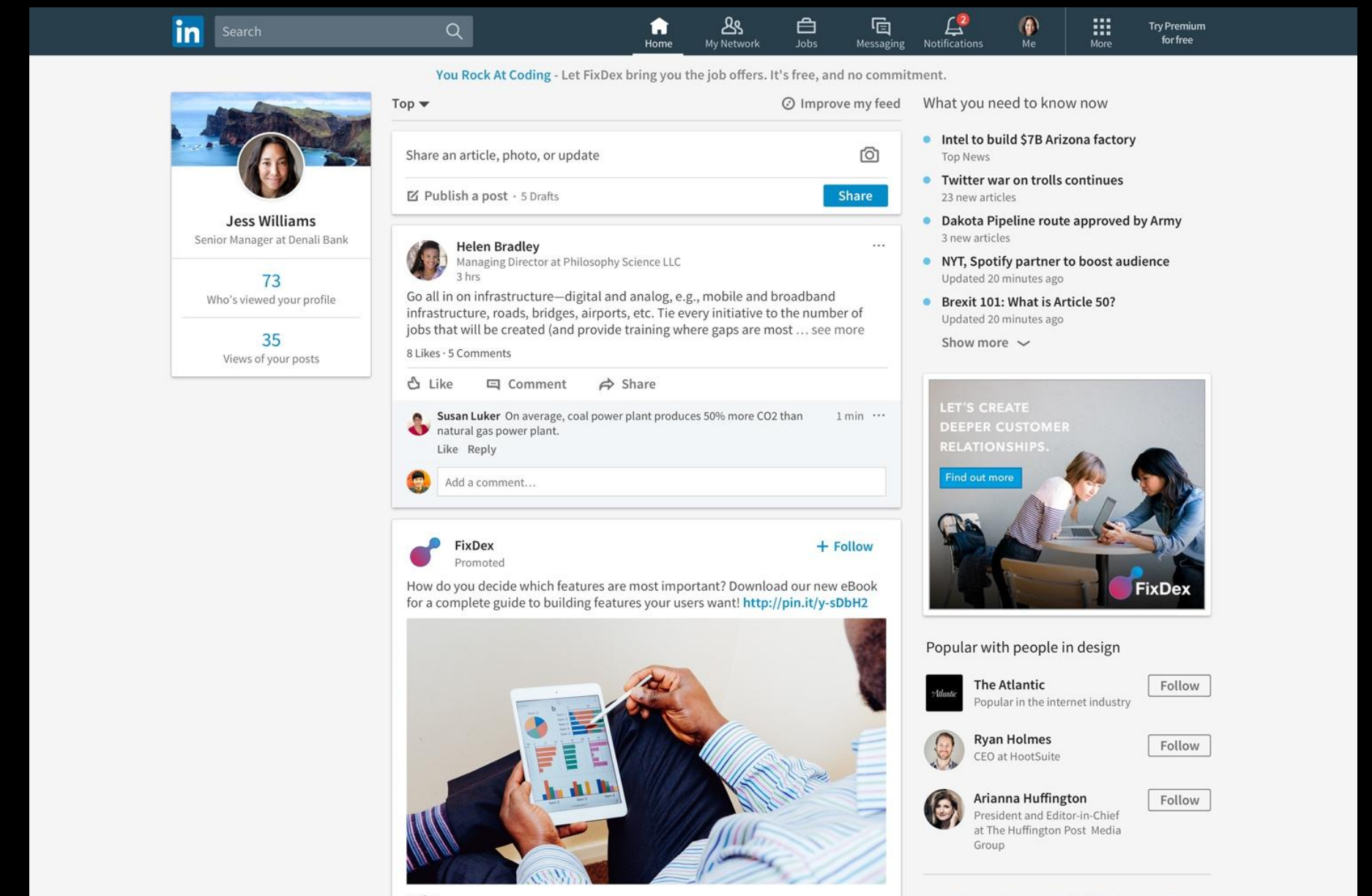
Mobile v/s Desktop



Feed on mobile

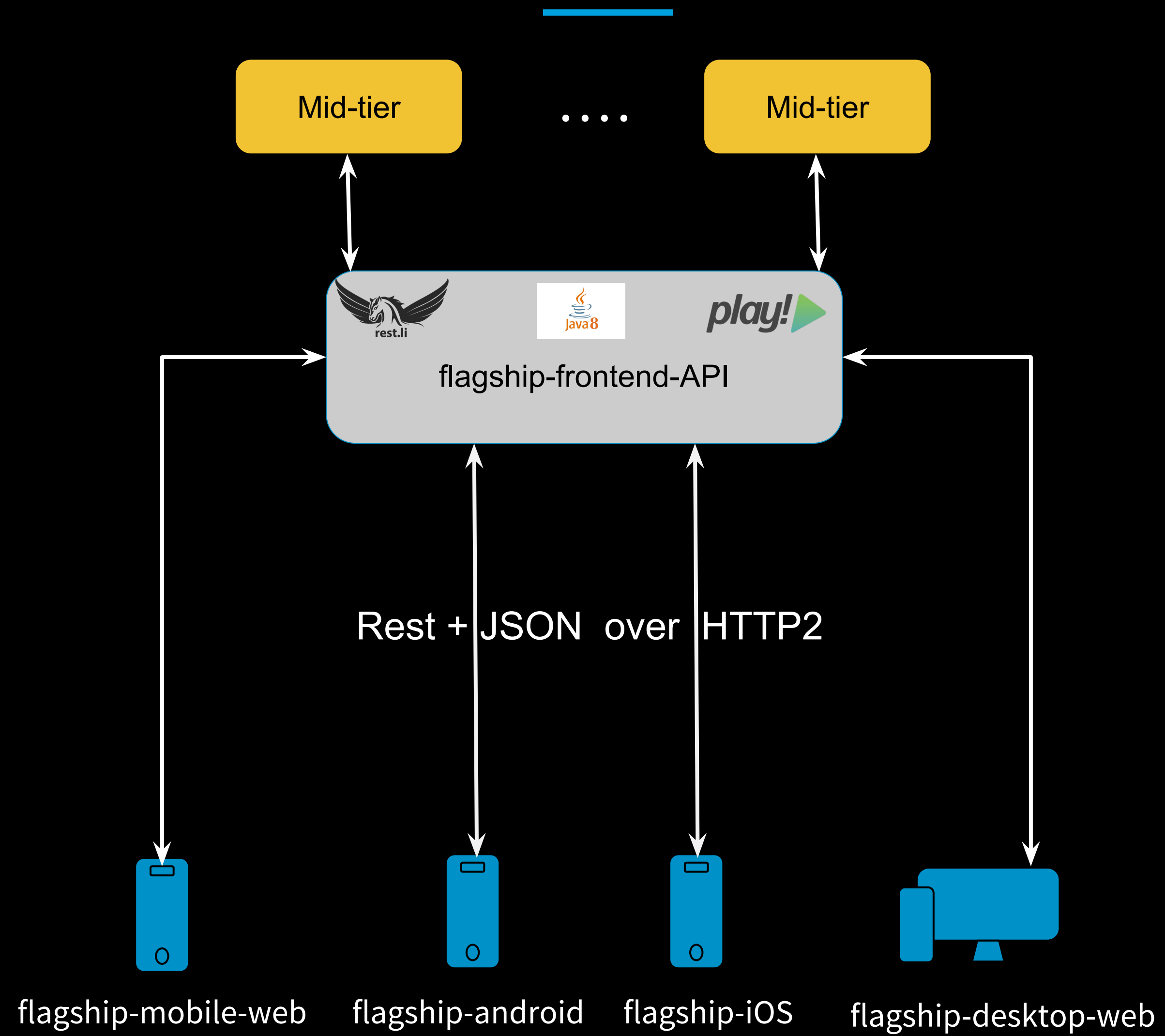


Feed on iPad



Feed on desktop

Client - Server setup



Scale

- > 120k QPS
- ~425 developers
- ~30 commits per day

3x3 Deployment

- Automated continuous release
- commit to production in < 3 hours
- 3 deployments a day

Modeling

Principles

- Backed by Strongly Typed Schemas
- Backward-compatible evolution
- No endpoint versioning

Schema definition

```
{
  "type": "record",
  "name": "TestRecord",
  "namespace": "com.linkedin.test",
  "doc": "Test",
  "fields": [
    {
      "name": "id",
      "type": "String",
      "doc": "ID"
    },
    {
      "name": "name",
      "type": "String",
      "doc": "Name",
      "optional": true
    }
  ]
}
```

iOS

```
@interface TestRecord : NSObject

@property(nonnull, copy) NSString *id;
@property(nullable, copy) NSString *name;

@end
```

Android

```
class TestRecord {

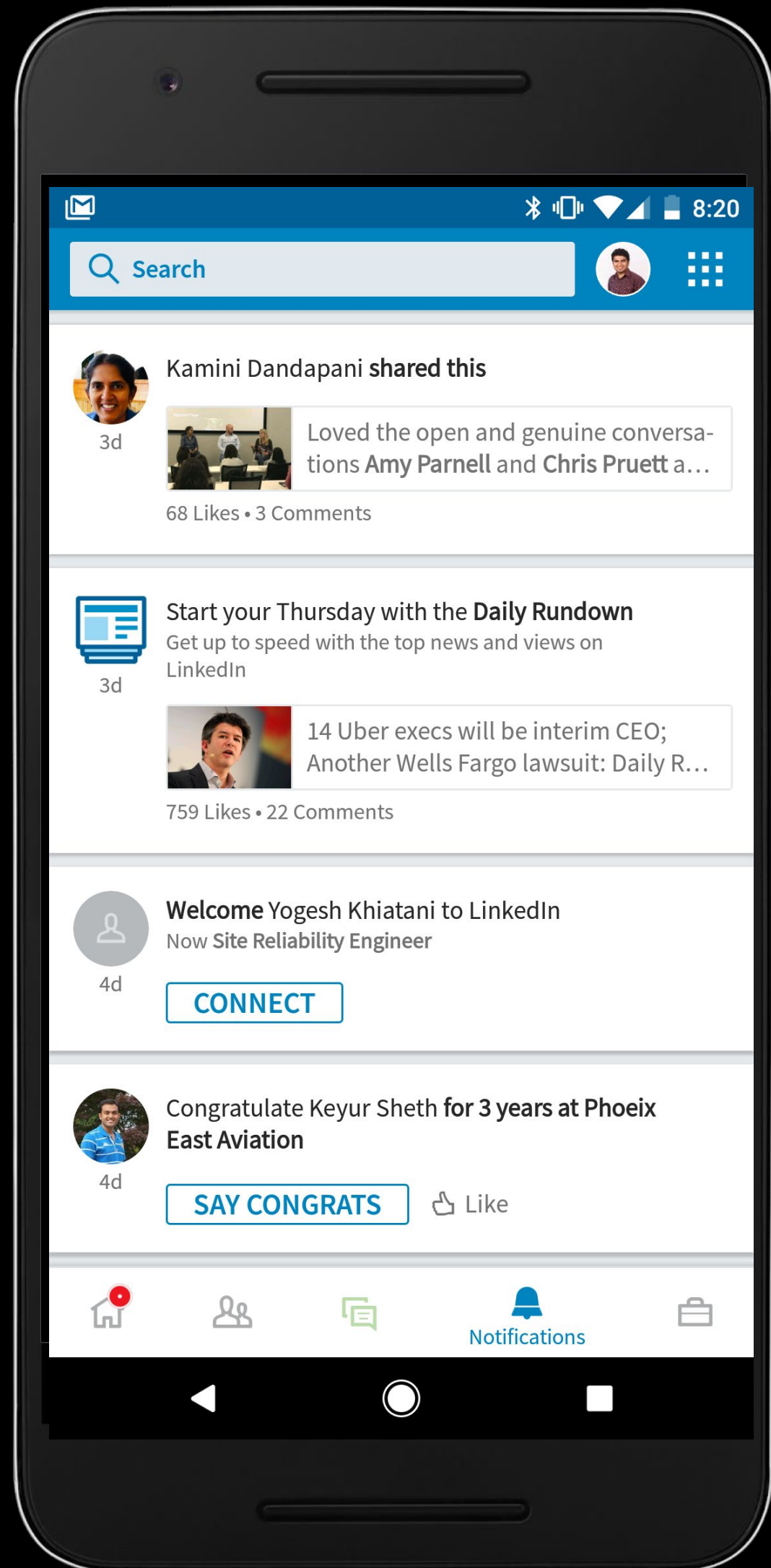
    @NonNull public final String id;
    @Nullable public final String name;

}
```

Web

```
export default DS.Model.extend({
  id: DS.attr('string'),
  name: DS.attr('string')
});
```

Entity Modeling

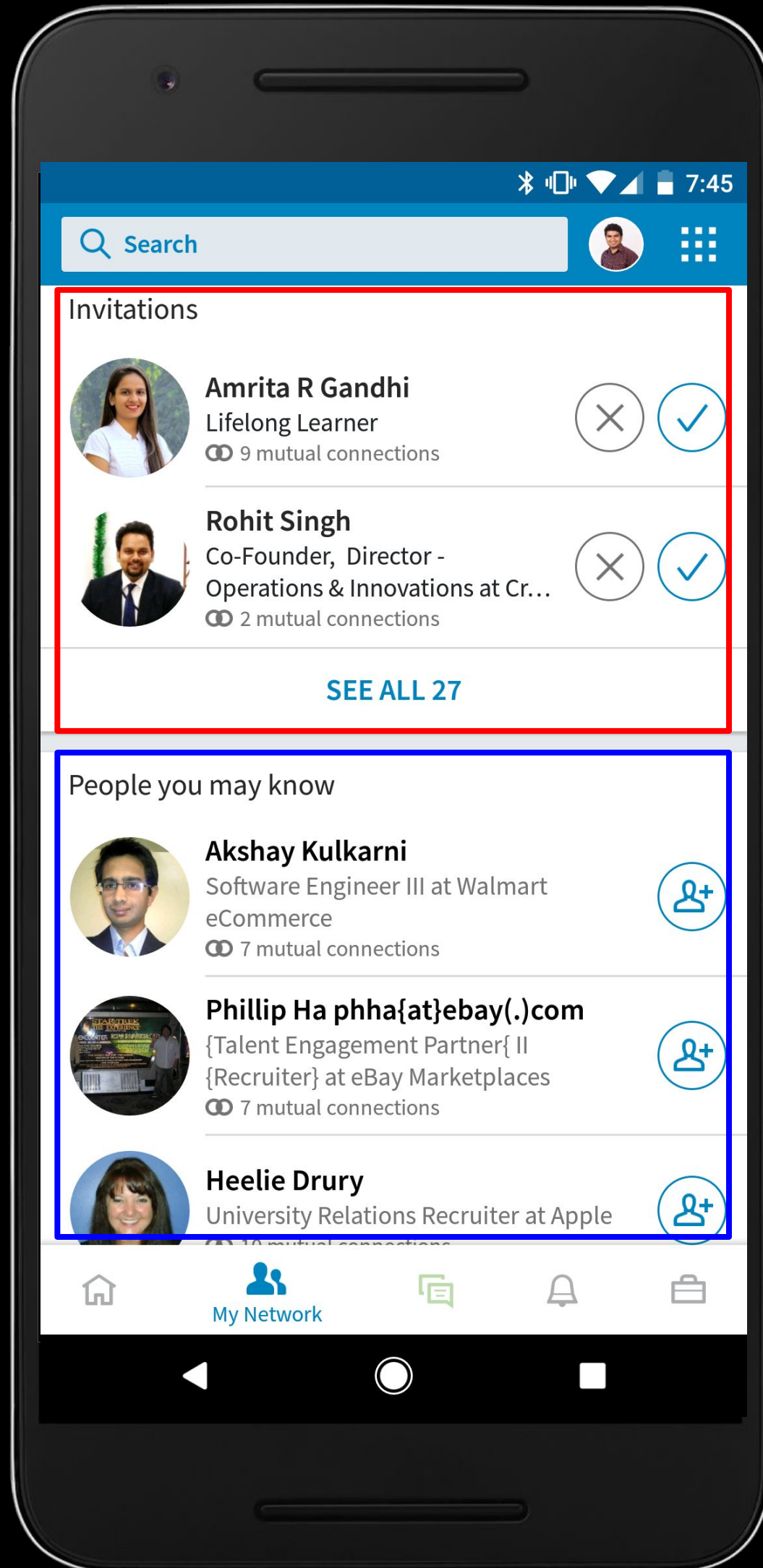


- Return - Collection<Card>

```
{
  "type": "record",
  "name": "Card",
  "namespace": "com.linkedin.voyager.identity.me",
  "doc": "A container for all the various cards used on the Me project.",
  "fields": [
    {
      "name": "entityUrn",
      "type": "com.linkedin.voyager.common.CardUrn",
      "doc": "Identifier for the card."
    },
    {
      "name": "value",
      "type": [
        "AggregateFollowCard",
        "AggregateGroupInvitationCard",
        "...",
        "CoursesYMBIINotificationCard"
      ]
    }
  ]
}

{
  "type": "record",
  "name": "AggregateFollowCard",
  "namespace": "com.linkedin.voyager.identity.me",
  "fields": [
    {
      "name": "publishedAt",
      "type": "com.linkedin.common.Time"
    },
    {
      "name": "followers",
      "type": {
        "type": "array",
        "items": "Profile"
      }
    },
    {
      "name": "numFollowers",
      "type": "long"
    },
    {
      "name": "read",
      "type": "boolean"
    }
  ]
}
```

Composite screens

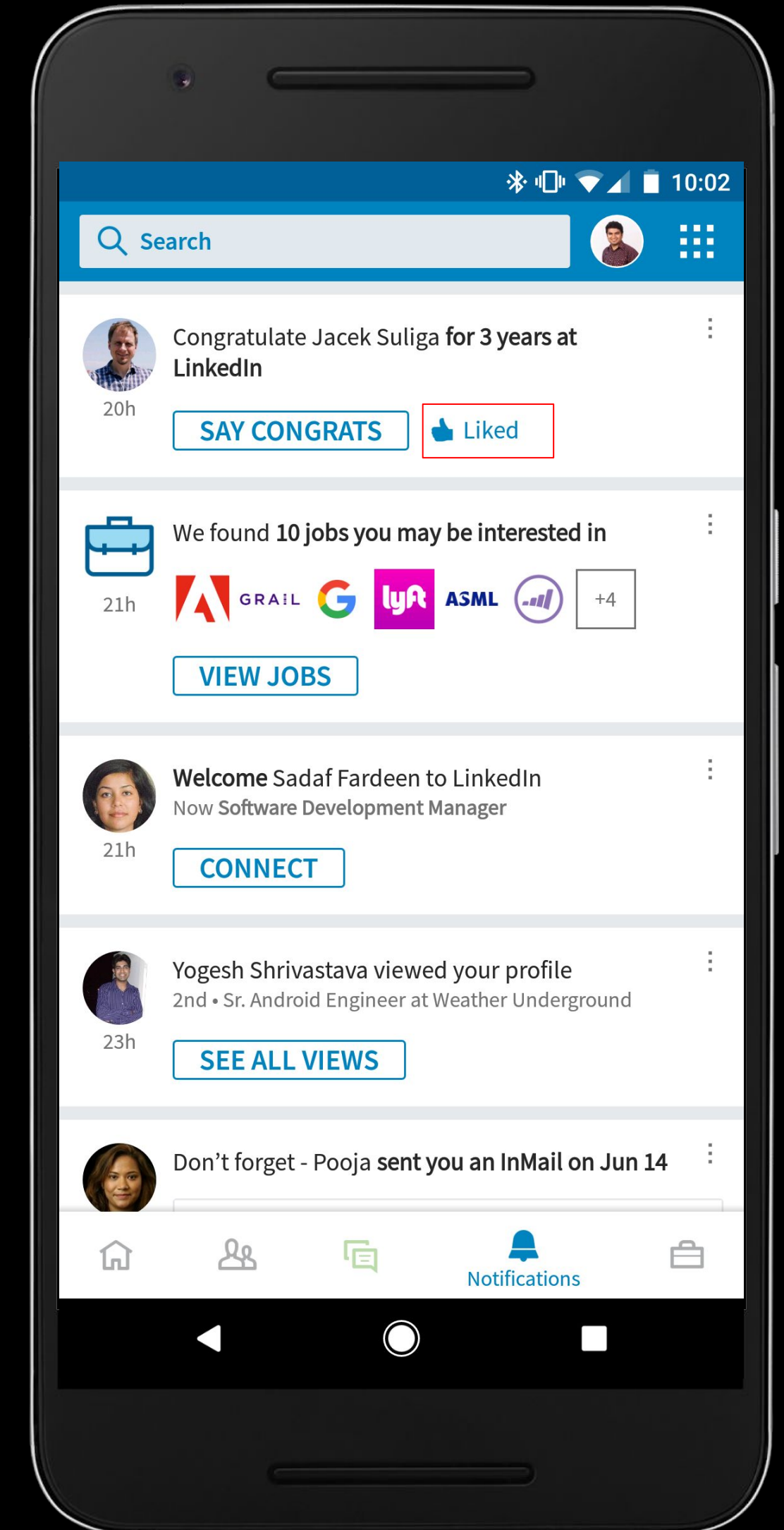
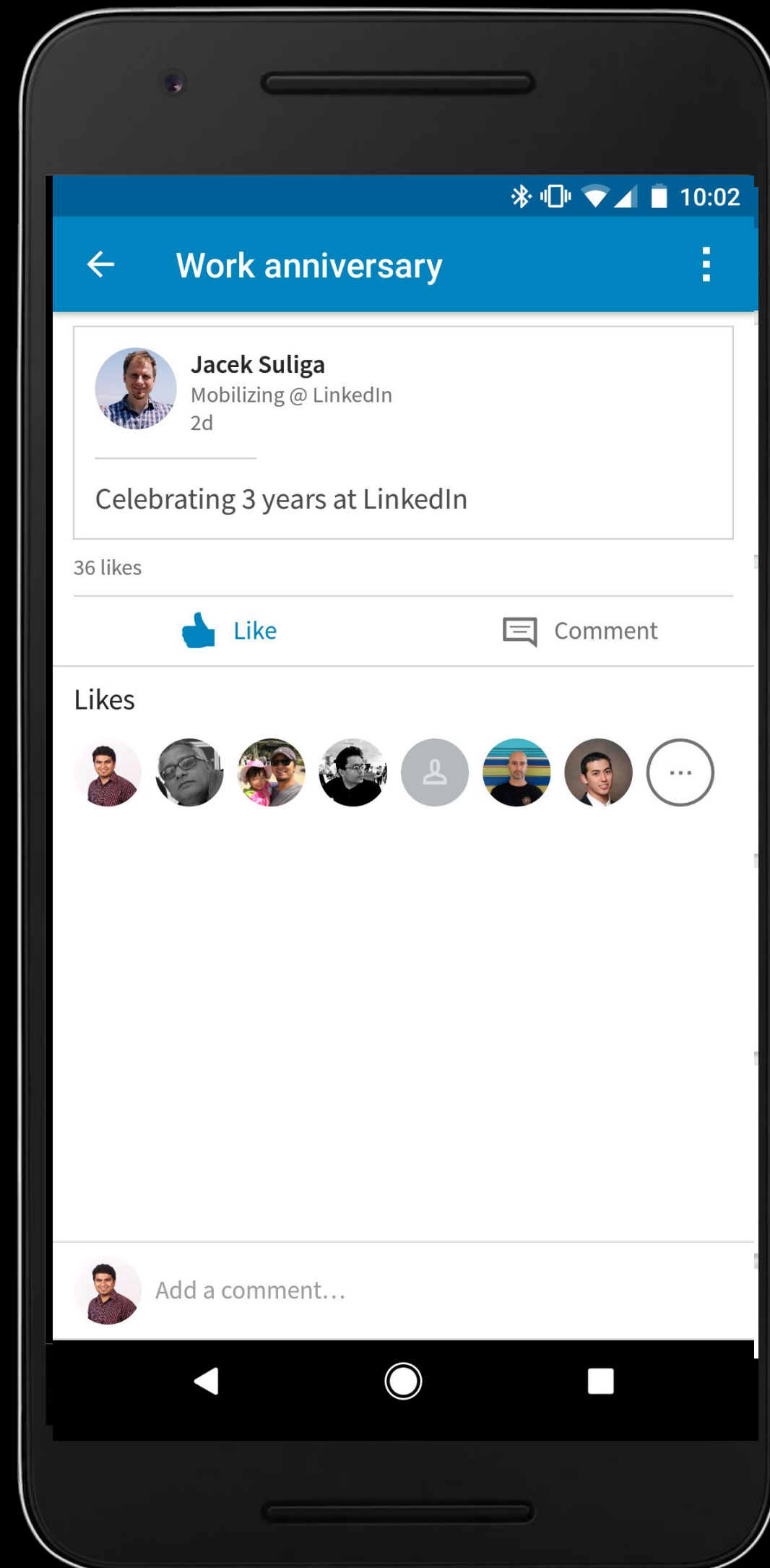
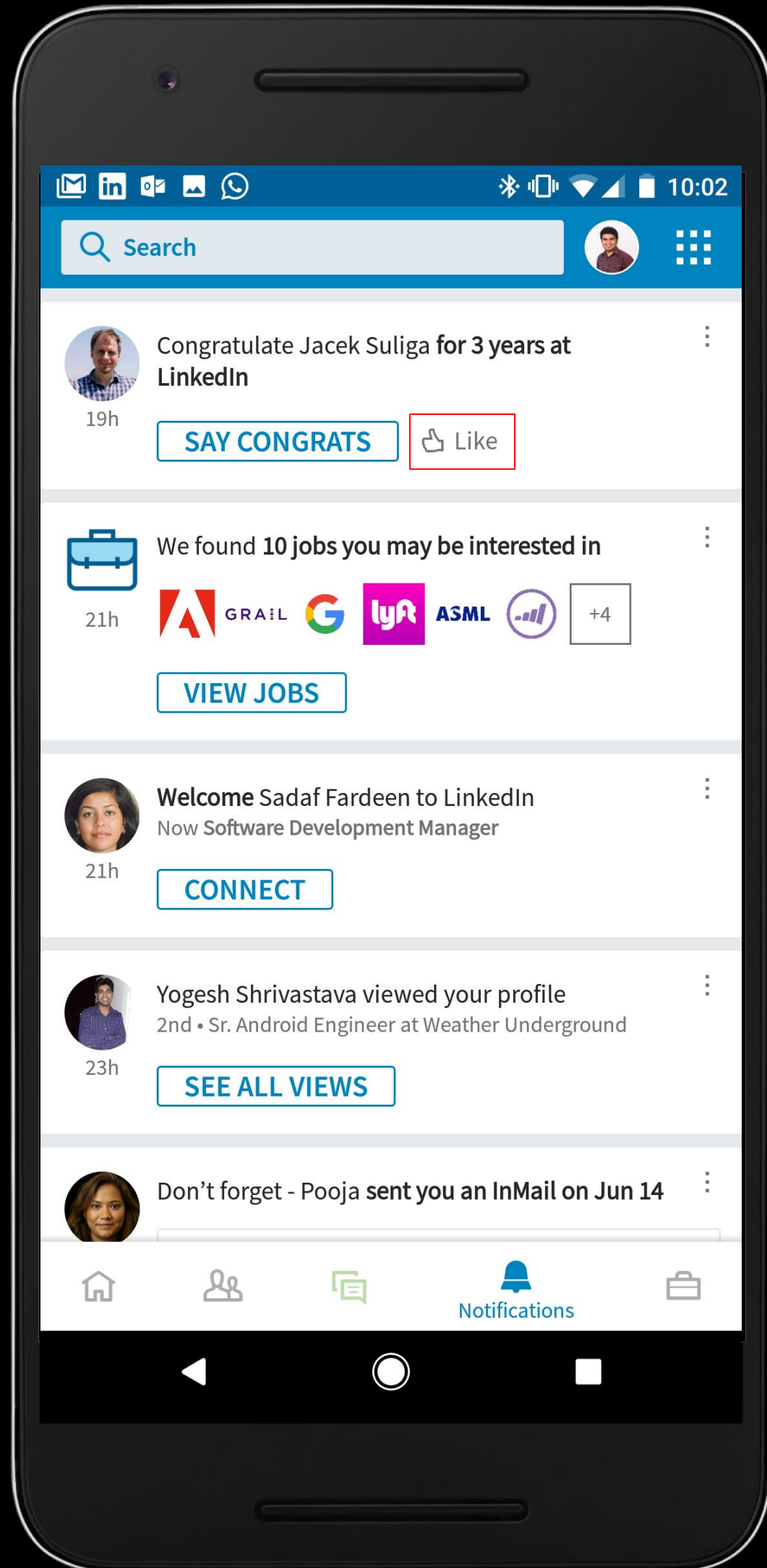


- Two top level resources
 - Invitations
 - PYMK (People You May Know)
- 1 network call to fetch both resources
 - Infrastructure level aggregation support

Advantages

- Easy to model
- Decouple API from UI
- Client side consistency

Client side consistency



Client side consistency

- **Why ?**
 - Good UX
 - Optimistic writes

Client side consistency

Can you do this auto-magically?

Client side consistency

Payload

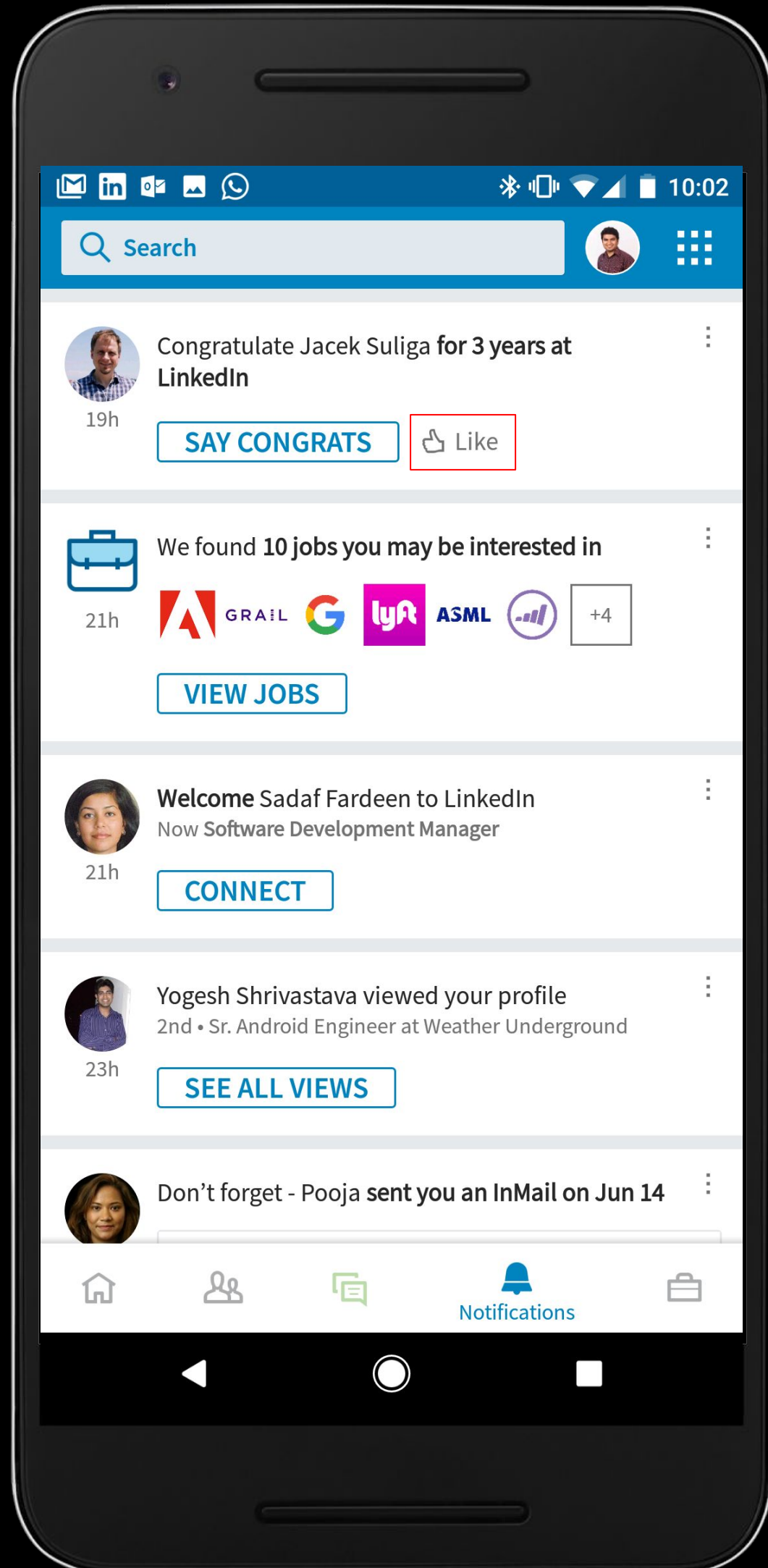
```
{
  "entityUrn": "urn:li:fs_notification:123",
  "publishedAt": 99999999,
  "update": {
    "entityUrn": "urn:li:fs_update:678",
    "type": "AnniversaryUpdate",
    "isLiked": false,
    ...
  },
  "miniProfile": {
    "entityUrn": "urn:li:fs_profile:456",
    "firstName": "Aditya",
    "lastName": "Modi",
    "pictureId": "/p/4/005/020/1e7/2f2fdcf.jpg",
    "headline": "Senior Software Engineer at LinkedIn"
  },
  ...
}
```

Cache

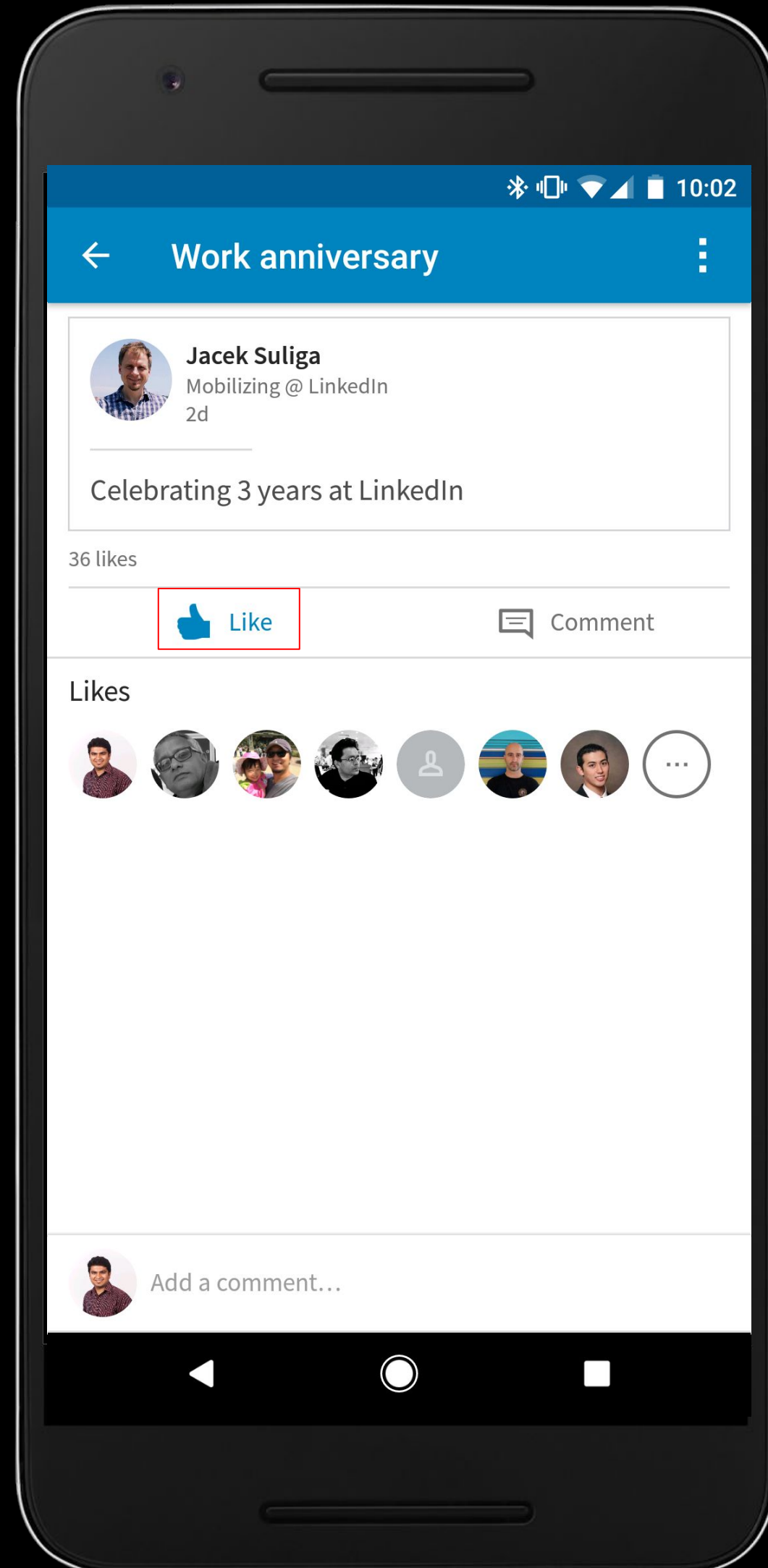
```
Key: urn:li:fs_notification:123
Value :
{
  "entityUrn": "urn:li:fs_notification:123",
  "publishedAt": 99999999,
  "update": {
    "uriStub": "urn:li:fs_update:678"
  },
  "miniProfile": {
    "uriStub": "urn:li:fs_profile:456",
  }
  ...
}

Key: urn:li:fs_update:678
Value :
{
  "entityUrn": "urn:li:fs_update:678",
  "type": "AnniversaryUpdate",
  "isLiked": false,
  ...
}

Key: urn:li:fs_profile:456
Value :
{
  ...
}
```



Client side consistency



Payload

```
{
  "entityUrn": "urn:li:fs_feed:123",
  "update": {
    "entityUrn": "urn:li:fs_update:678",
    "type": "AnniversaryUpdate",
    "isLiked": true,
    ...
  },
  "socialSummary": {
    "entityUrn": "urn:li:fs_socialSummary:456",
    "numLikes": 100,
    "numComments": "20",
    ....
  },
  ...
}
```

Cache

```
Key: urn:li:fs_feed:123
Value :
{
  "entityUrn": "urn:li:fs_feed:123",
  "update": {
    "uriStub": "urn:li:fs_update:678",
  },
  "socialSummary": {
    "uriStub": "urn:li:fs_socialSummary:456",
  },
  ...
}

Key: urn:li:fs_update:678
Value :
{
  "entityUrn": "urn:li:fs_update:678",
  "type": "AnniversaryUpdate",
  "isLiked": true,
  ...
}

Key: urn:li:fs_socialSummary:456
Value :
{
  ...
}
```


Client side consistency

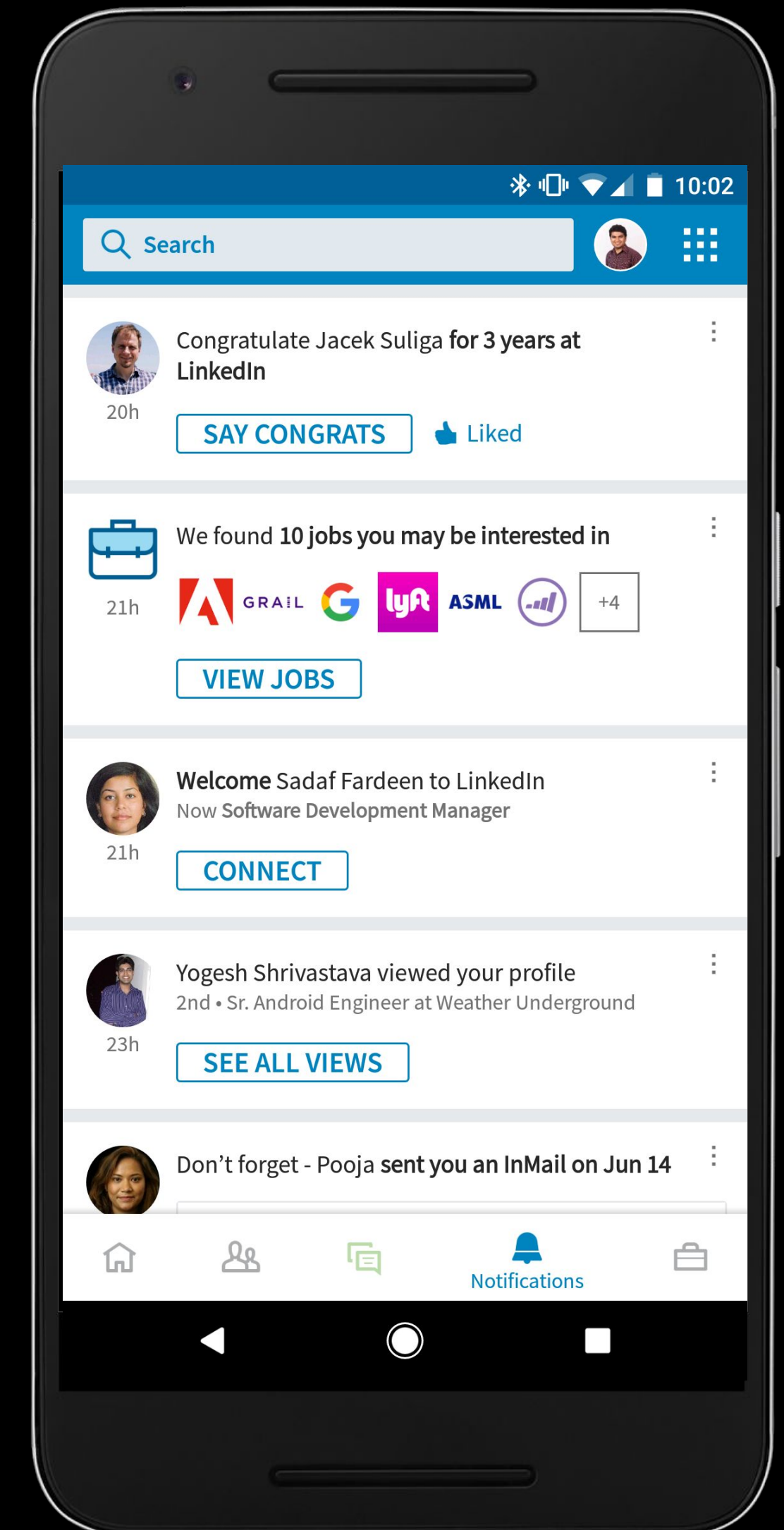
Cache

```
Key: urn:li:fs_notification:123
Value :
{
  "entityUrn":"urn:li:fs_notification:123",
  "publishedAt": 9999999
  "update": {
    "uriStub":"urn:li:fs_update:678"
  },
  "miniProfile":{
    "uriStub":"urn:li:fs_profile:456",
  }
  ...
}

Key: urn:li:fs_update:678
Value :
{
  "entityUrn":"urn:li:fs_update:678",
  "type": "AnniversaryUpdate",
  "isLiked": true,
  ...
}
```

Payload

```
{
  "entityUrn":"urn:li:fs_notification:123",
  "publishedAt": 9999999
  "update": {
    "entityUrn":"urn:li:fs_update:678",
    "type": "AnniversaryUpdate",
    "isLiked": true,
    ...
  },
  "miniProfile":{
    "entityUrn":"urn:li:fs_profile:456",
    "firstName":"Aditya",
    "lastName":"Modi",
    "pictureId":"/p/4/005/020/1e7/2f2fdcf.jpg",
    "headline":"Senior Software Engineer at LinkedIn"
  },
  ...
}
```





Everything is awesome, right?

Takes a long time to ship a feature

Use case: Introduce a new kind of notification



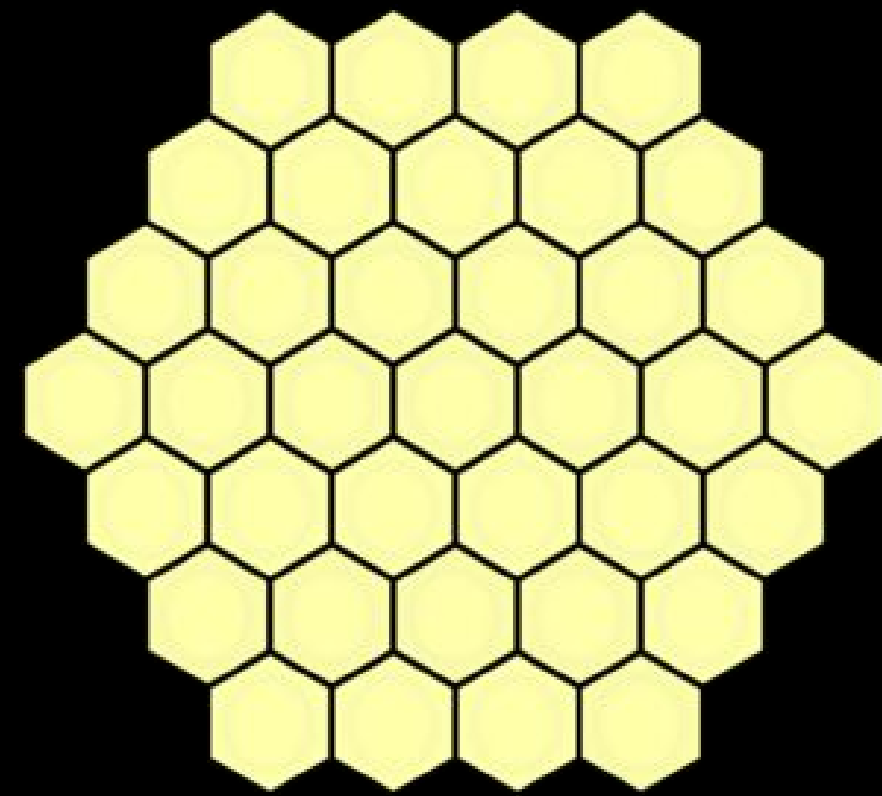
Why so long?

- Create new models for every feature
- Write code on each client platform
- Wait for native app release/adoption

Challenge

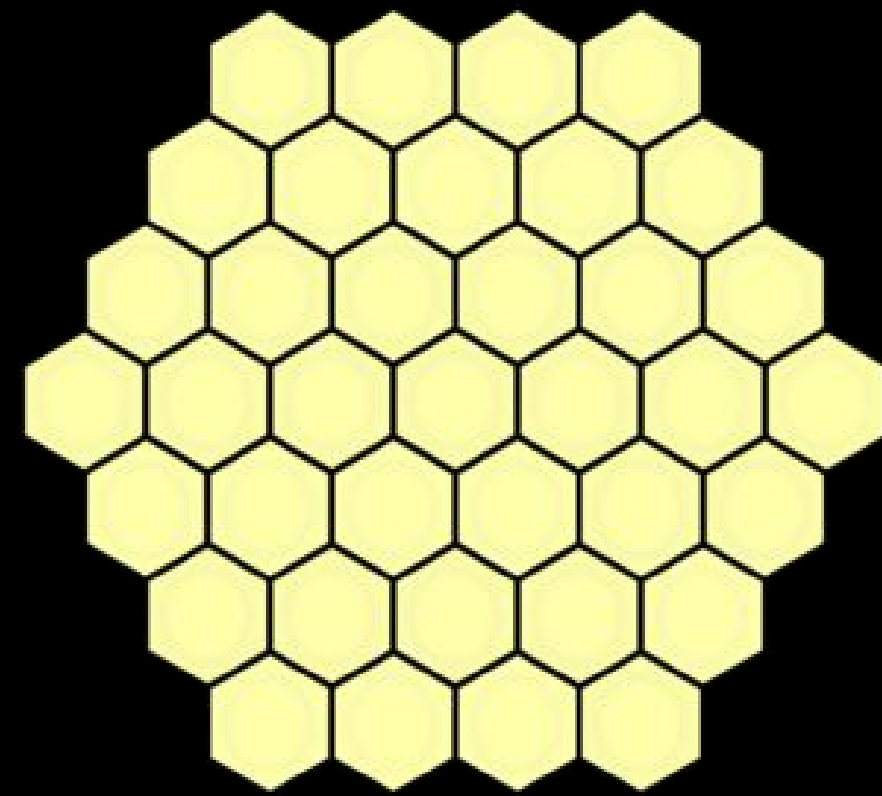
Cut this down to 1 day!

Project Honeycomb



- Quickly build and release notifications
- Increase user engagement
- Sweet and sticky, just like honey!

Beyond iteration speed...

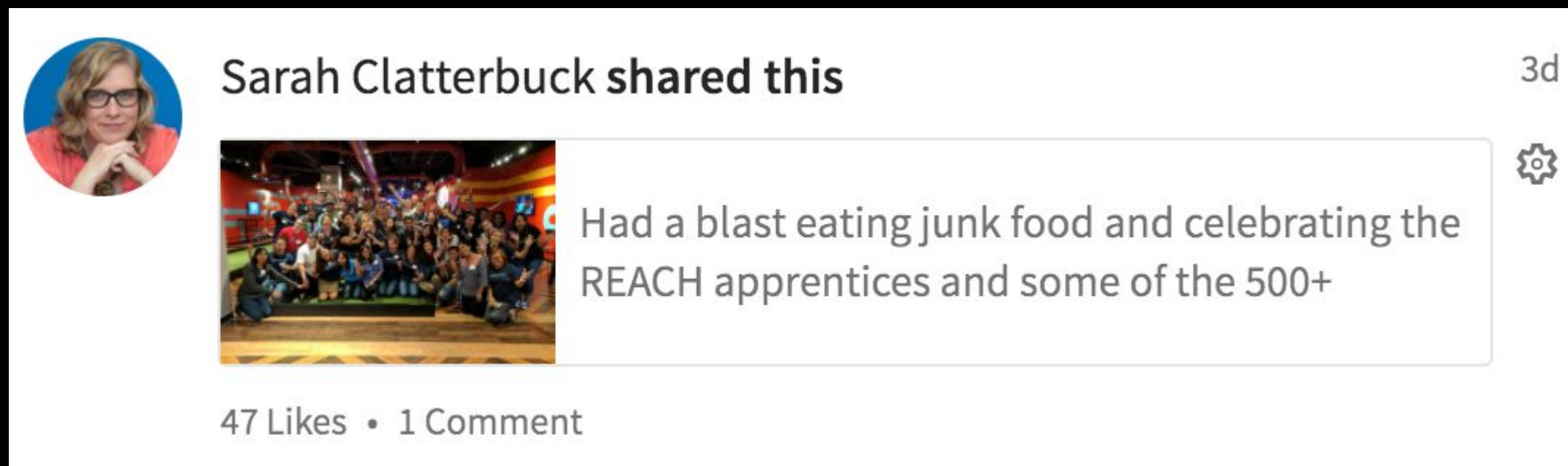


- New notifications WITHOUT app updates
- Client side consistency
- Stellar runtime performance

View Template API

- Model based on how the UI view looks
- Similar views grouped into 1 template
- More UI specific logic on API server

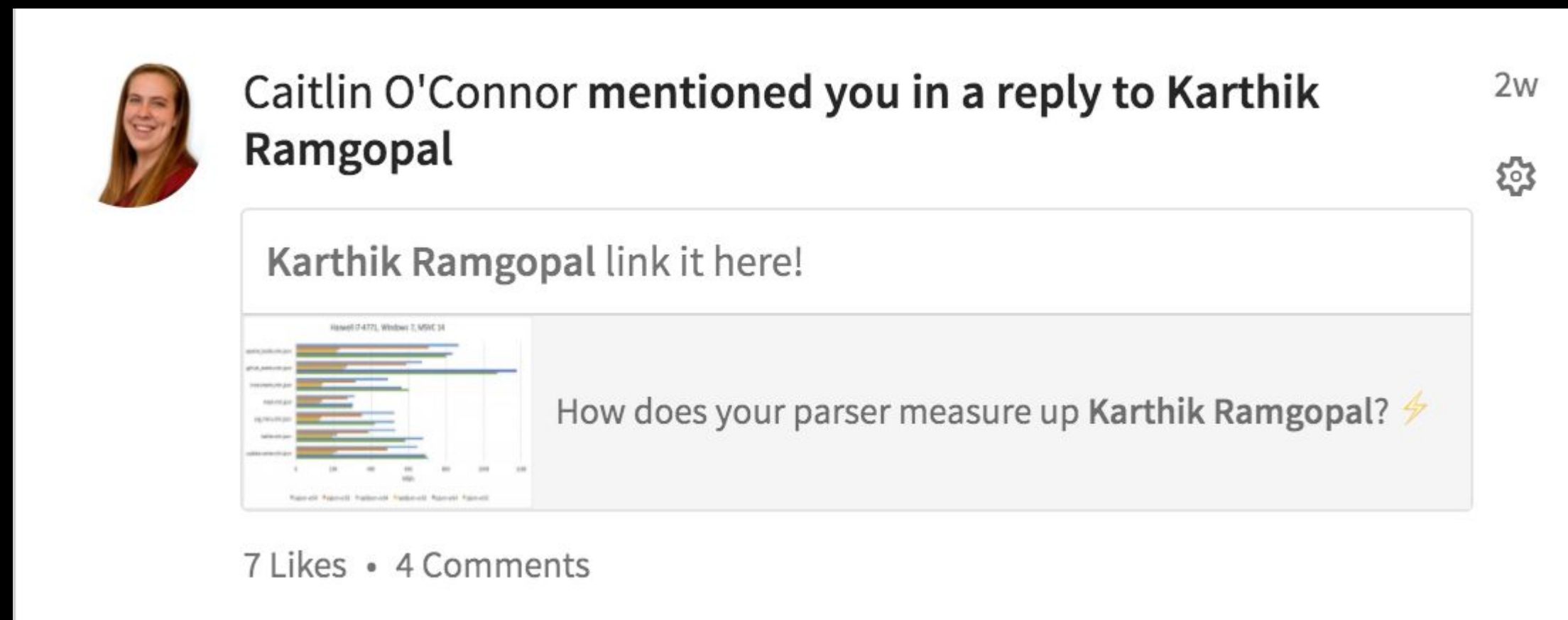
Share notification



Share Template

- PrimaryImage: URL?
- Title: AttributedString
- Timestamp: Long
- ShareImage: URL?
- ShareTitle: String
- LikeCount: Long? (Default: 0)
- CommentCount: Long? (Default: 0)

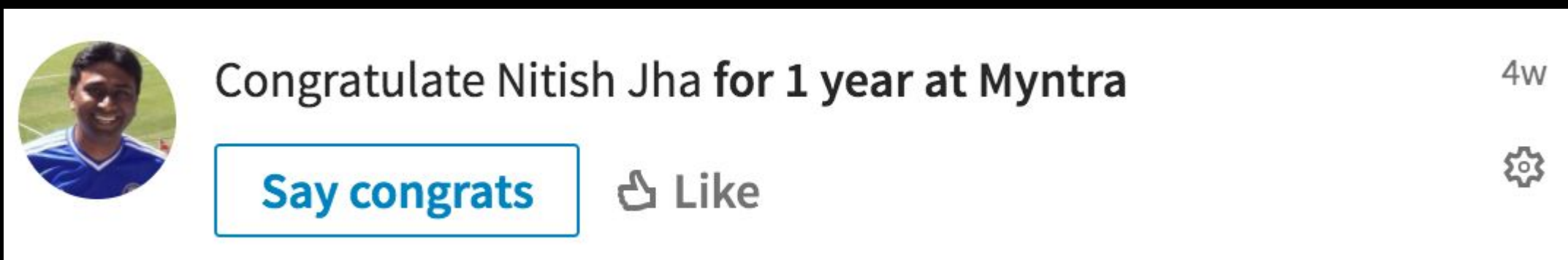
Now let's look at a slightly different notification



Modify Share Template

- PrimaryImage: URL?
- Title: AttributedString
- Timestamp: Long
- ShareImage: URL?
- ShareTitle: ~~String~~ AttributedString
- ShareSubtitle: AttributedString?
- LikeCount: Long? (Default: 0)
- CommentCount: Long? (Default: 0)


How about something radically different?



Work Anniversary Template


- PrimaryImage: URL?
- Title: AttributedString
- Timestamp: Long


Something slightly different again?



Congratulate Laurie Zieman for starting a new position as **Sourcing Specialist at Novo Nordisk** 4w

Previously Senior Talent Advisor, People Science for Facebook at R4R
(Recruiting 4 Recruiters) @Facebook/People Science

[Say congrats](#)  Like



Work Anniversary/**New Position** Template

- PrimaryImage: URL?
- Title: AttributedString
- Timestamp: Long
- **BodyText: AttributedString?**

How do we return a heterogeneous list?

- Use Rest.li **paginated collections**. Differentiate between items using a **Union**.
- JSON payload structure:

```
{  
  "elements" : [  
    {"Share": {"Title": "Sarah Clatterbuck shared a...", ...}},  
    {"Anniversary": {"Title": "Congratulate Nitish Jha...", ...}},  
    ....  
  ],  
  "paging": { "start" : 0, "count": 10, "paginationToken": "xydsad"}  
}
```

Minor payload optimization

- Embed the type into the payload to reduce nesting.
- JSON payload structure:

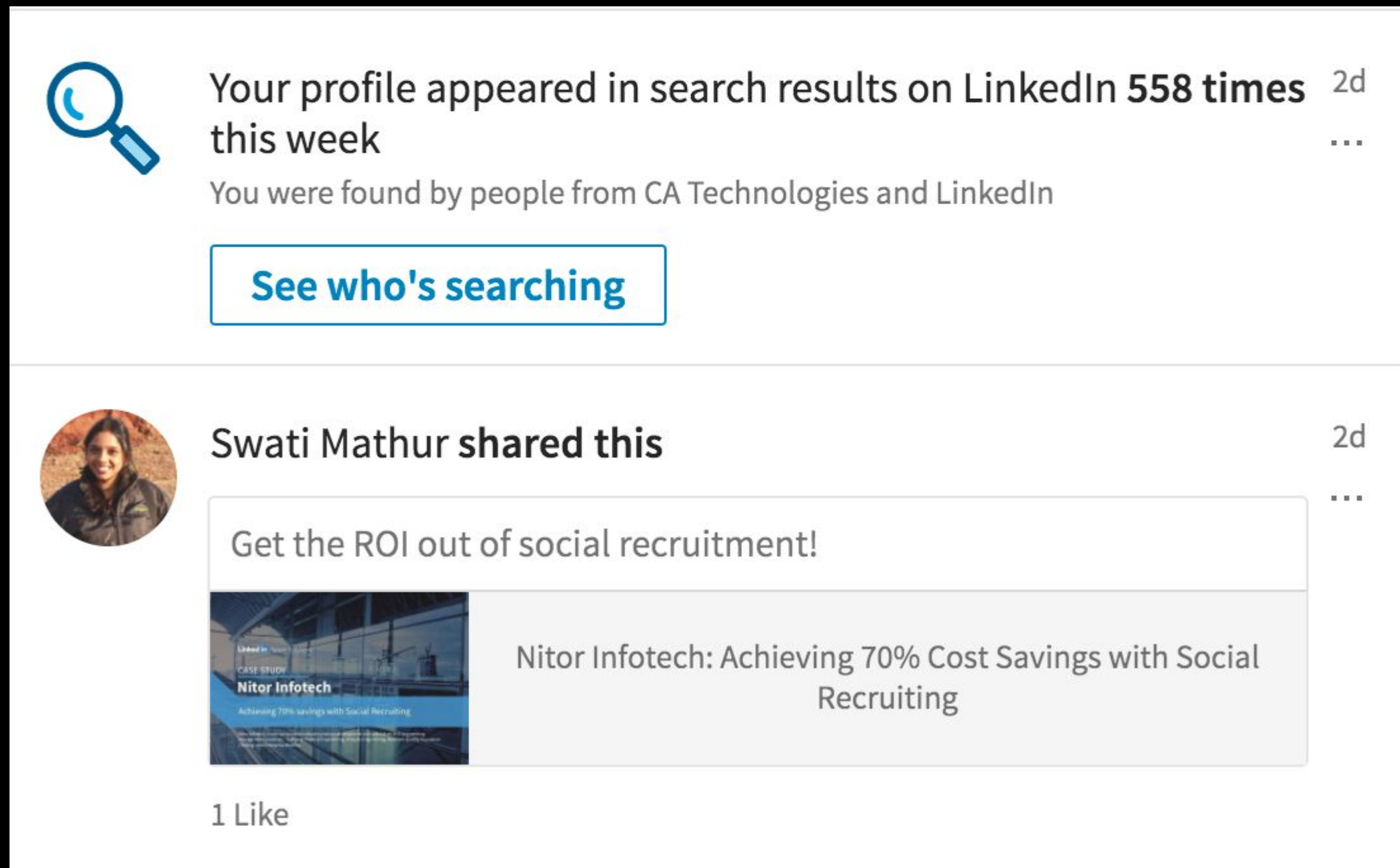
```
{  
  "elements" : [  
    {"Type": "Share", "Title": "Sarah Clatterbuck shared a...", ...},  
    {"Type": "Anniversary", "Title": "Congratulate Nitish Jha...", ...},  
    ....  
  ],  
  "paging": { "start" : 0, "count": 10, "paginationToken": "xydsad"}  
}
```

Client side rendering

- Code-generated response parser
- Bind model to native views
- Write once* per layout, reuse.

Backward compatibility

Drop unknown notification types.




The screenshot shows two notification items. The first item, titled "Your profile appeared in search results on LinkedIn 558 times this week", includes a magnifying glass icon, a sub-header "You were found by people from CA Technologies and LinkedIn", and a button labeled "See who's searching". The second item, titled "Swati Mathur shared this", features a profile picture of Swati Mathur, a text snippet "Get the ROI out of social recruitment!", and a card for "Nitor Infotech: Achieving 70% Cost Savings with Social Recruiting" which includes a small image of a building and the Nitor Infotech logo. Below the card, it shows "1 Like".


```
{  
  "elements" : [  
    {"Stat": {"Title": "Your Profile...", ...}},  
    {"JYMBII": {"Title": "5 Jobs you", ...}},  
    {"Share": {"Title": "Swati Mathur...", ...}},  
    ....  
  ],  
  "paging": { "start" : 0, "count": 10,  
    "paginationToken": "xydsad"}  
}
```




Backward compatibility

Drop unknown fields based on product needs.




Congratulate Laurie Zieman for starting a new position as **Sourcing Specialist at Novo Nordisk** 4w


[Say congrats](#)  Like



Congratulate Laurie Zieman for starting a new position as **Sourcing Specialist at Novo Nordisk** 4w

Previously Senior Talent Advisor, People Science for Facebook at R4R (Recruiting 4 Recruiters) @Facebook/People Science

[Say congrats](#)  Like



Benefits

- New notification types without client changes
- Renders faster on the client



But... Client side Consistency is lost!

How do we solve this?

**“DON'T REINVENT THE WHEEL, JUST REALIGN
IT.”**

ANTHONY J. D'ANGELO

© Lifehack Quotes

How did we solve the `AttributedString` problem?

`AttributedString`

- Model formatted text
- Control formatting from the server
- Impractical to use HTML

AttributedString schema

AttributedString

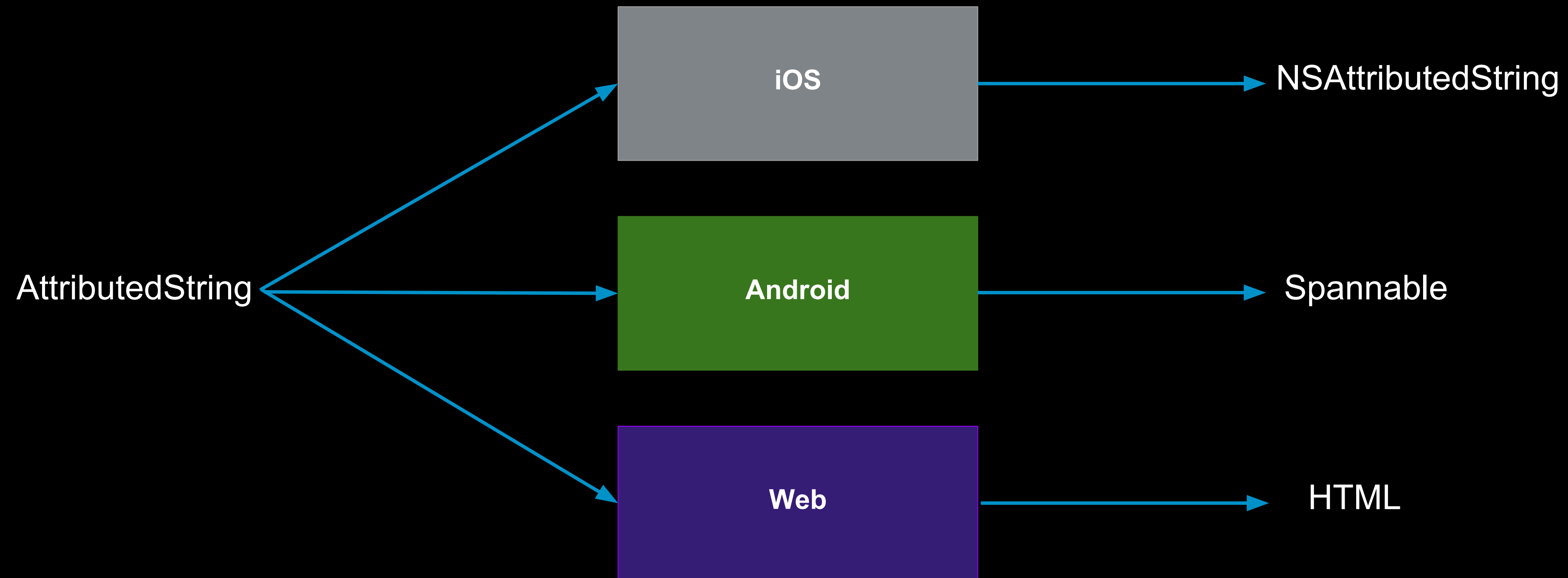
- Text: String
- Attributes: List[Attribute]  BOLD, ITALIC, UNDERLINE etc.

Attribute

- Type: AttributeType
- StartIndex: Int
- Length: Int
- Metadata: Any?

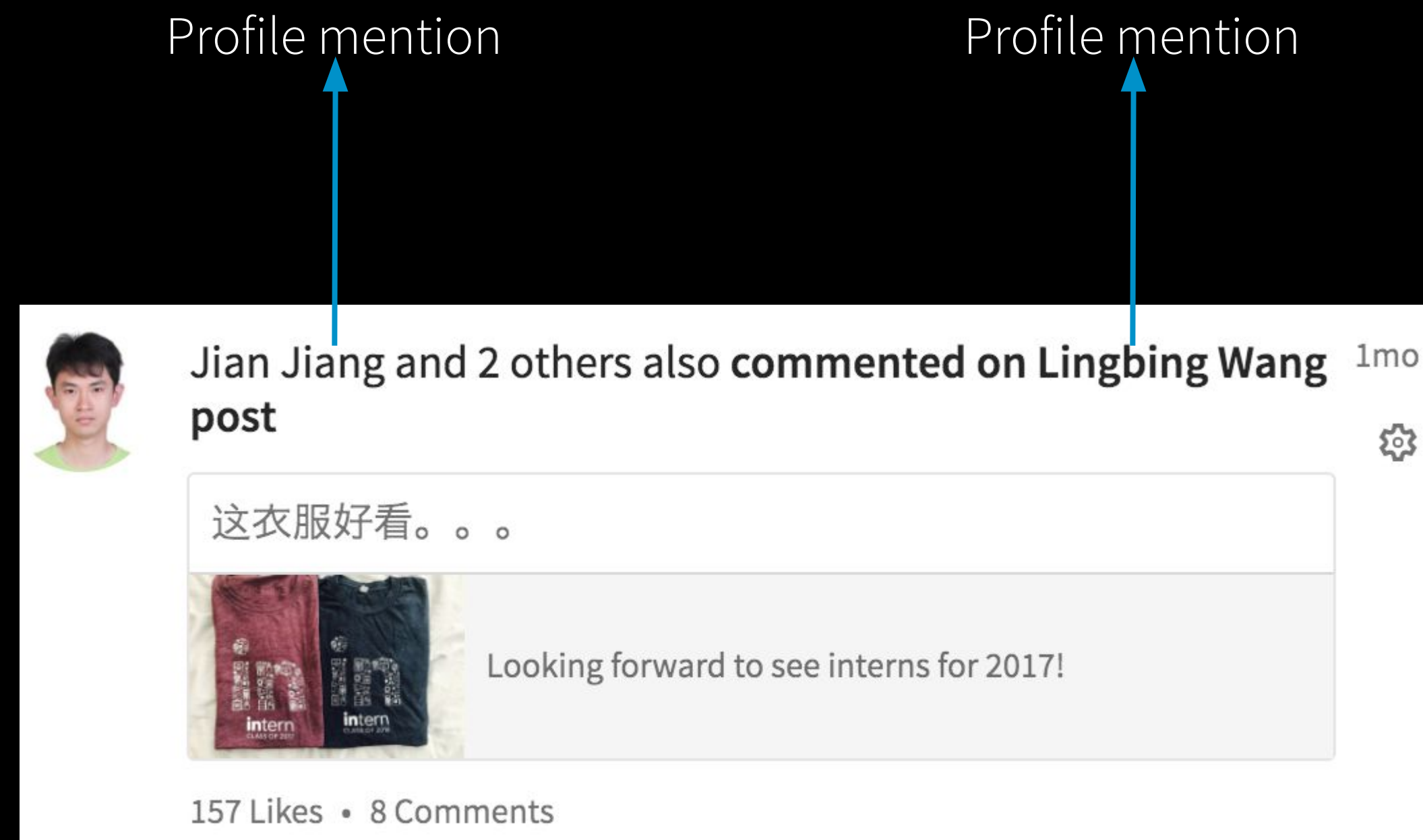
Platform specific binding

Infrastructure provided support



What if we extended this concept to entity mentions?

Model entity mentions also as a custom formatting specifier.



Introducing TextViewModel

TextViewModel

- Text: String
- Attributes: List

TextViewAttribute

- Type: TextViewAttributeType
- StartIndex: Int
- Length: Int
- Metadata: Any?

● Profile: Profile?

● Job: Job?

● Company: Company?

● Course: Course?

Similar to AttributedString

Flattened canonical entities as optional fields

Entity mentions

Entities could be mentioned in different ways.

The image displays two social media posts on a dark background. The first post on the left features a profile picture of a man and the text "Don't forget - Eric sent you an InMail on Jun 14" with a "2d" timestamp. Below this is a white-bordered box containing the text "InMail • Adobe iOS Development Opportunity - Can we chat?" and a "Reply now" button. A blue arrow points from the text "First Name" above to the name "Eric" in the post. The second post on the right features a profile picture of a man and the text "Congratulate Sumit Agarwal for starting a new position as Deputy Manager at APCER Life Sciences" with a "3d" timestamp. Below this is a white-bordered box containing the text "Previously Assistant Manager at APCER Life Sciences", a "Say congrats" button, and a "Like" button. Two blue arrows point from the text "Full Name" and "Position" above to the name "Sumit Agarwal" and the job title "Deputy Manager at APCER Life Sciences" respectively.

TextViewAttributeType

TextViewAttributeType

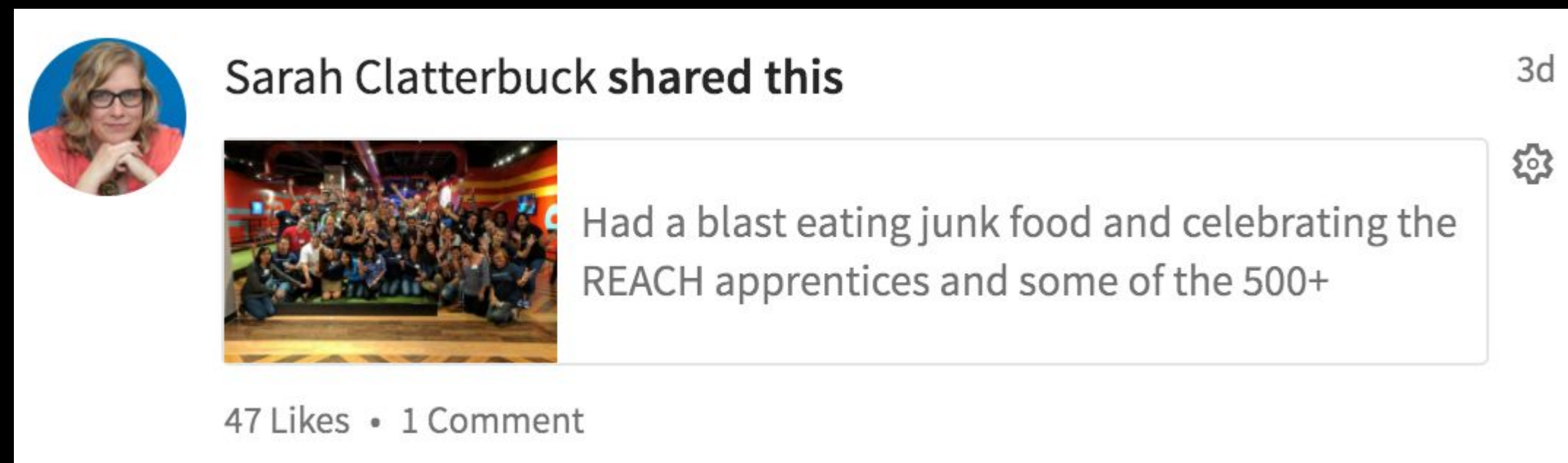
- PROFILE_FIRST_NAME
- PROFILE_FULL_NAME
- PROFILE_HEADLINE
- PROFILE_DISTANCE
- COMPANY_NAME
- COMPANY_HEADLINE
- JOB_TITLE
-

If a particular type is used, then the corresponding entity is populated by the server.

- PROFILE_XXX types will populate the profile field for example with the corresponding profile.

Backward compatibility++

Old clients cannot handle new mention types. **Always send Raw text though redundant.**



```
{
  "title" : {
    "Text": "Sarah Clatterbuck shared this",
    "Attributes": [
      {"Type": "PROFILE_FULL_NAME",
        "StartIndex": 0....}
    ]
  }
}
```

Watch Out

- Singular and Plurals
- Possessive forms
- i10n and i18n


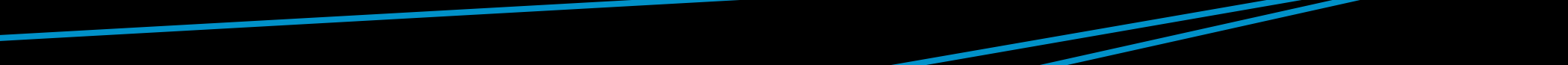
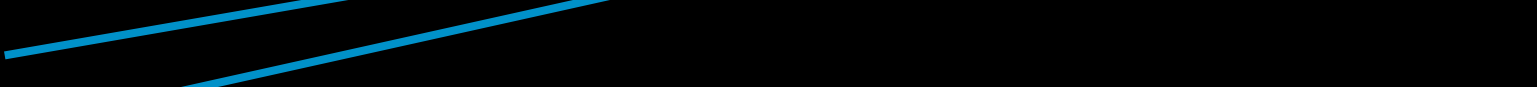
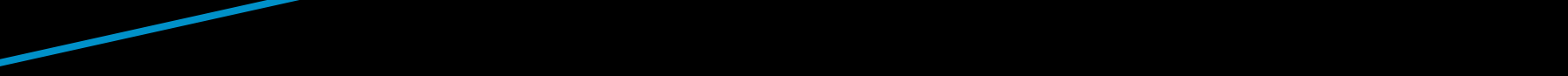
How about images?

Use the same concept as **TextViewModel**. Introduce **ImageViewModel**.

ImageViewModel

- Attributes: List[ImageViewAttribute]

ImageViewAttribute

- ImageViewAttributeType
 - URL: URL?
 - ResourceName: String?
 - Profile: Profile? 
 - Job: Job? 
 - Company: Company? 
 - Course: Course? 
- Flattened canonical entities as optional fields

ImageViewAttributeType

ImageViewAttributeType

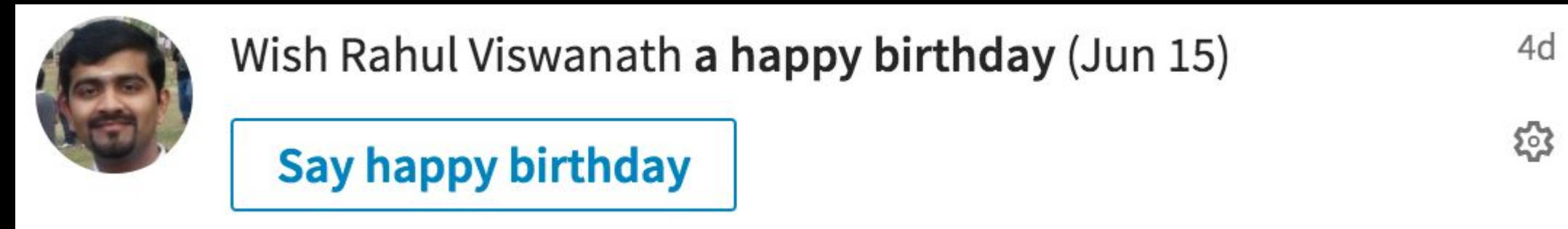
- URL
- RESOURCE_NAME
- PROFILE_IMAGE
- PROFILE_BACKGROUND
- COMPANY_IMAGE
-

If a particular type is used, then the corresponding entity is populated by the server.

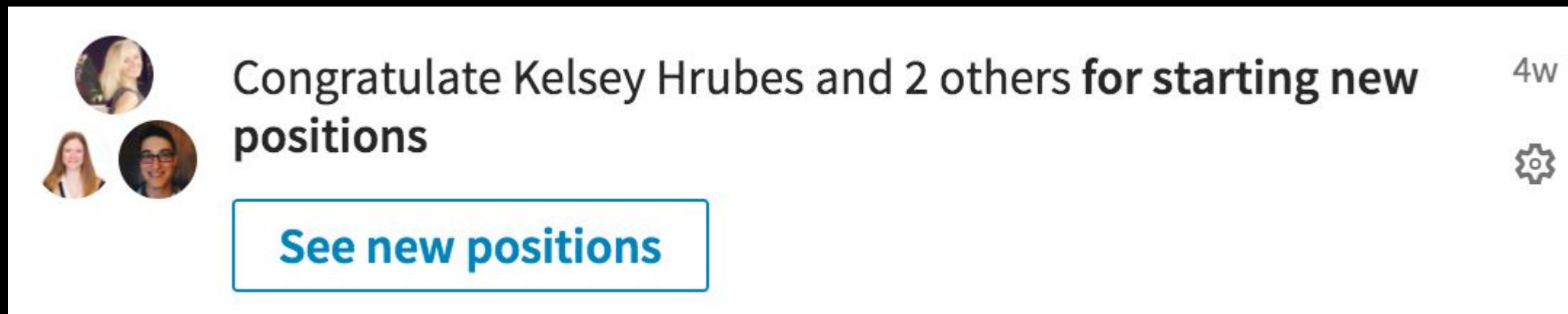
- PROFILE_XXX types will populate the profile field for example with the corresponding profile.
- URL type will populate the image URL
- RESOURCE_NAME will populate the pre-canned resource name.

Rendering Images

- Infra code extracts Image URL out of **ImageViewModel**
- Load into platform specific image view.



One Attribute: Regular ImageView



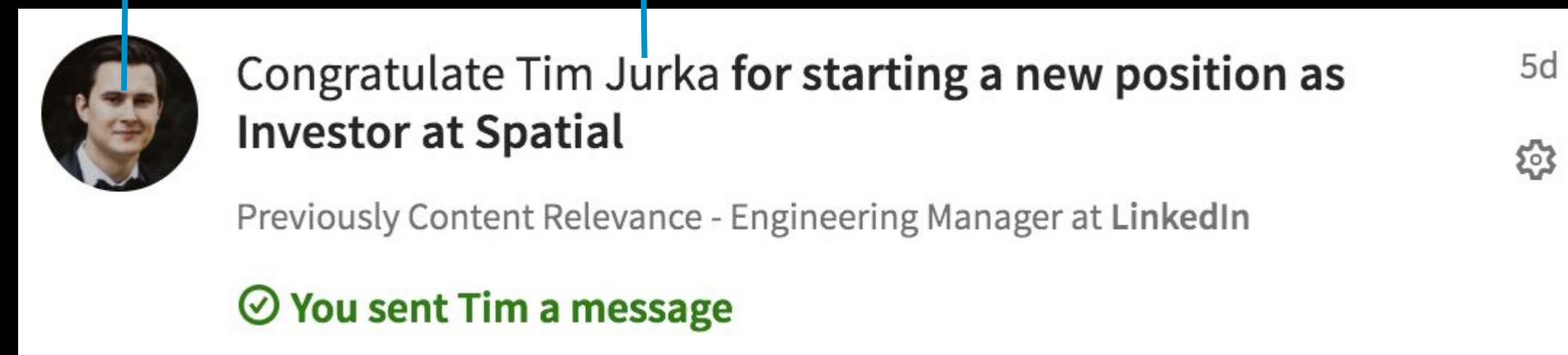
Multiple Attributes: **GridImageView**

Performance considerations

Entities may repeat multiple times within the same notification causing payload size bloat.

Tim's Profile in ImageViewModel

Tim's Profile in TextViewModel



Solution: Response Normalization

All canonical entities have a unique ID. Use a JSON API like response format.

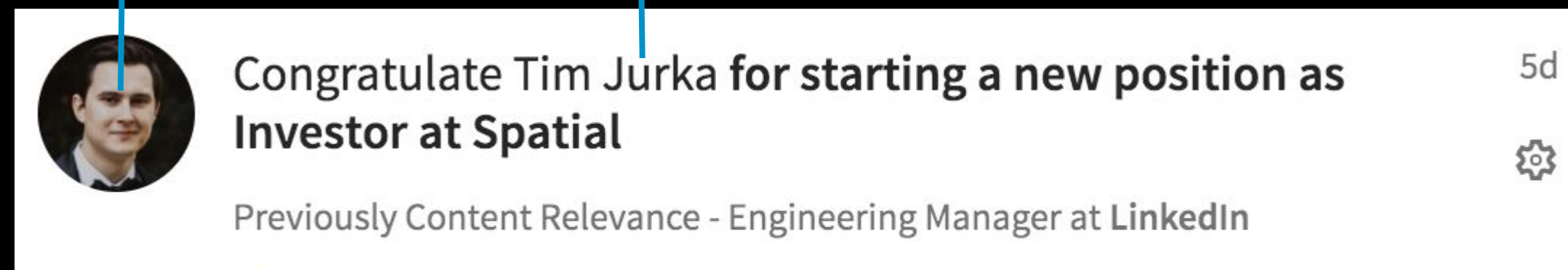
```
{
  "data" : {
    "Profile": "profile:123", ...
  },
  "included": [
    {
      "id" : "profile:123", "firstName" : "Tim", "LastName" : "Jurka", ... }
    },
    ....
  ]
}
```

Performance considerations (Continued)

All fields from the entities may not be used.

ImageURL

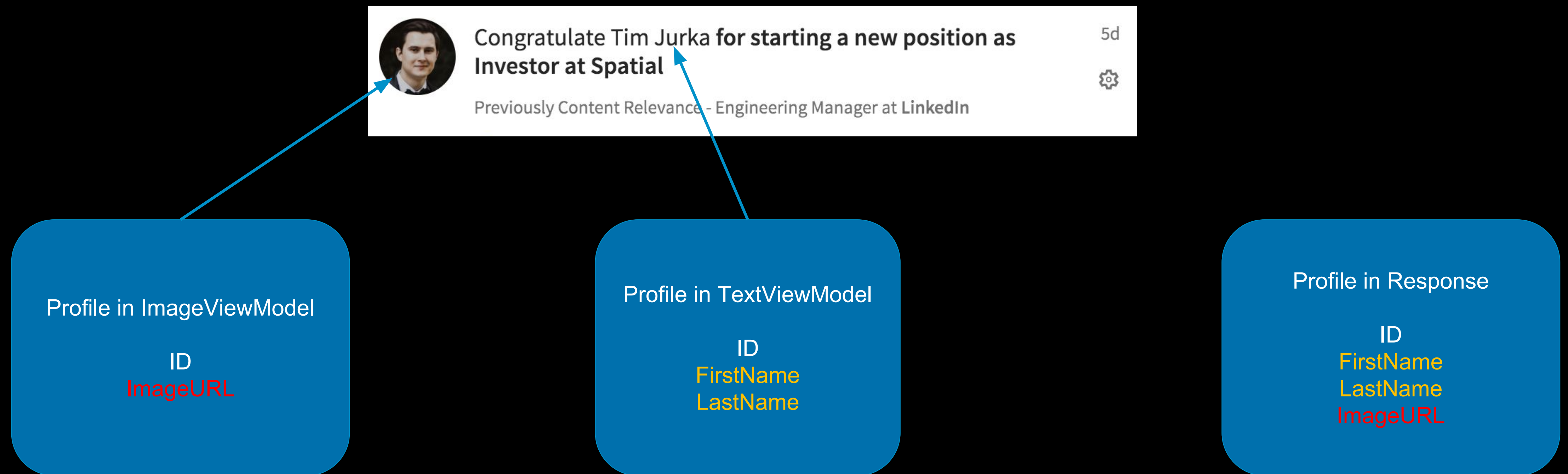
First Name and Last Name



A screenshot of a LinkedIn post snippet. On the left is a circular profile picture of a man. To the right of the picture is the text: "Congratulate Tim Jurka for starting a new position as Investor at Spatial". Below this is the text: "Previously Content Relevance - Engineering Manager at LinkedIn". On the far right of the snippet, there is a "5d" timestamp and a gear icon for settings.

Solution: Deco

Deco is a LinkedIn framework that allows selective projection and decoration of fields.





Results

Improved Developer and Product Velocity

9 new notification types in all of 2016

16 new notification types in May and June 2017

API model reduction

42 API models for old UI

6 API models for new UI

Code size reduction

15k LOC app and test code for old UI

3k LOC app and test code for new UI

Future Direction

- Extend to other pages in Flagship
- Extend to other LinkedIn apps like JobSeeker, Recruiter etc.



Out of scope for this talk

- Intricate details of client side consistency management
- Generic Modeling of actions
- Challenges in migrating from the old notifications API to the new one

Find us after the talk, and we're happy to chat about this and more.

Q & A

Aditya Modi

amodi@linkedin.com

<https://www.linkedin.com/in/modiaditya/>

Karthik Ramgopal

kramgopal@linkedin.com

<https://www.linkedin.com/in/karthikrg/>

