Service-to-service Communication Protocols.

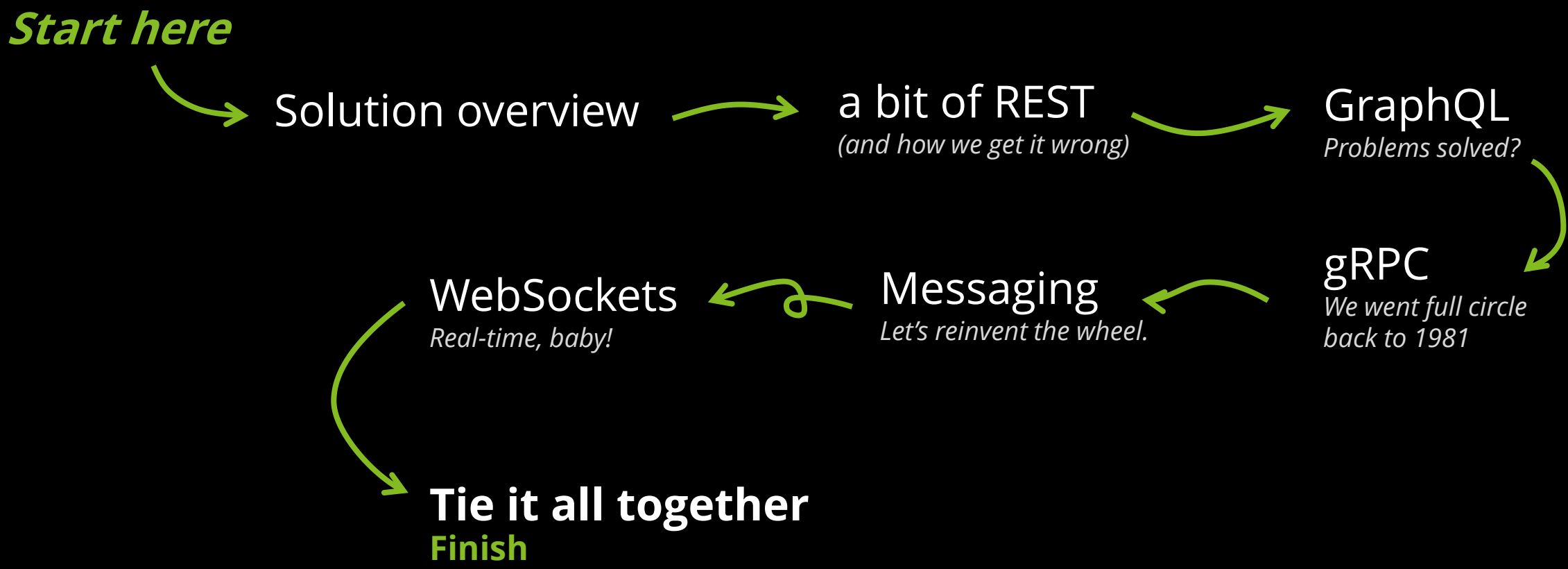
Nicklas Millard

Software Development Manager in Deloitte Consulting.

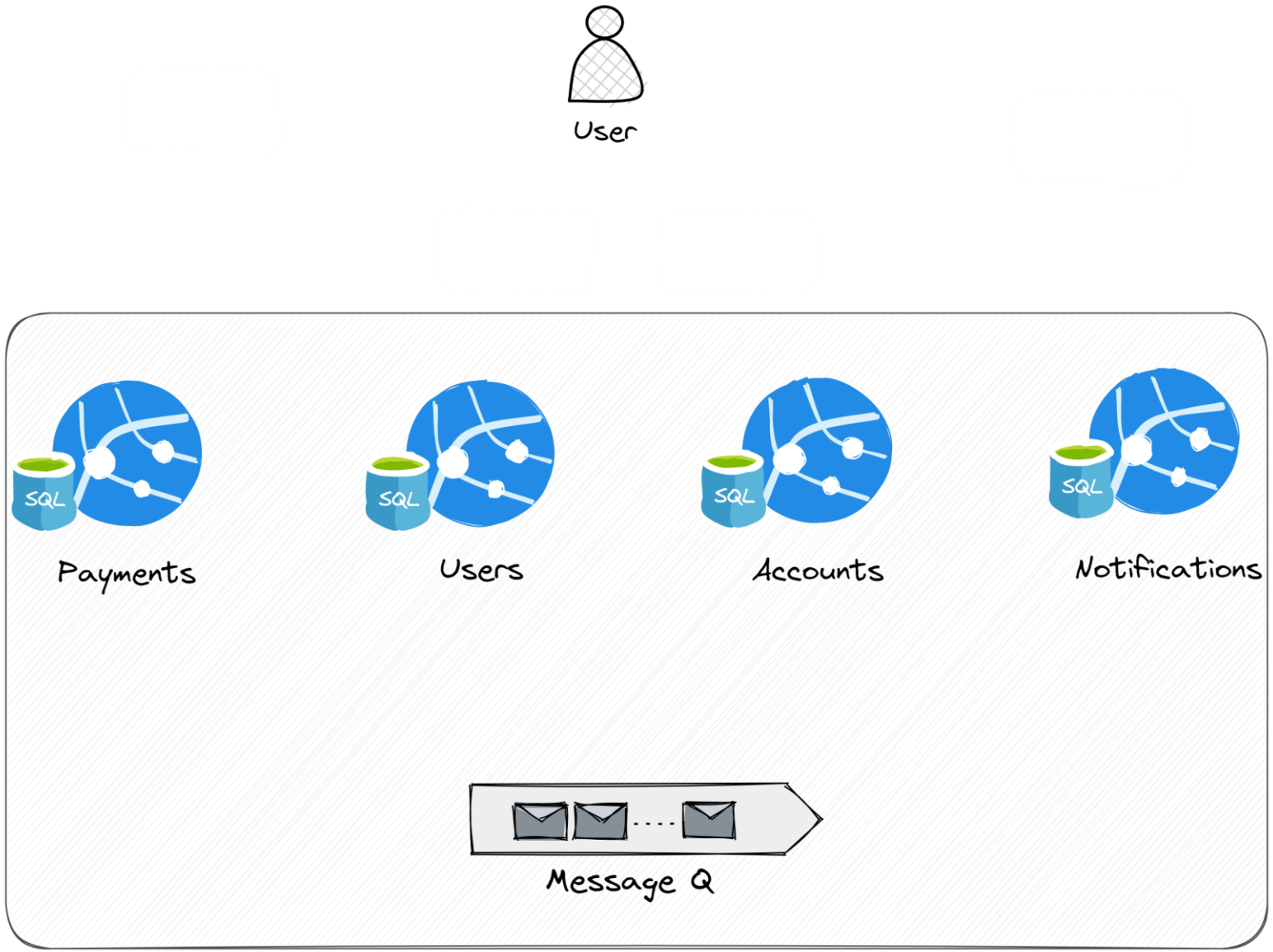
- Public and Private clients
- From solo developer to 20 person projects
- Done ~17 application development projects for 15 clients.
- FinTech industry experience

Roadmap

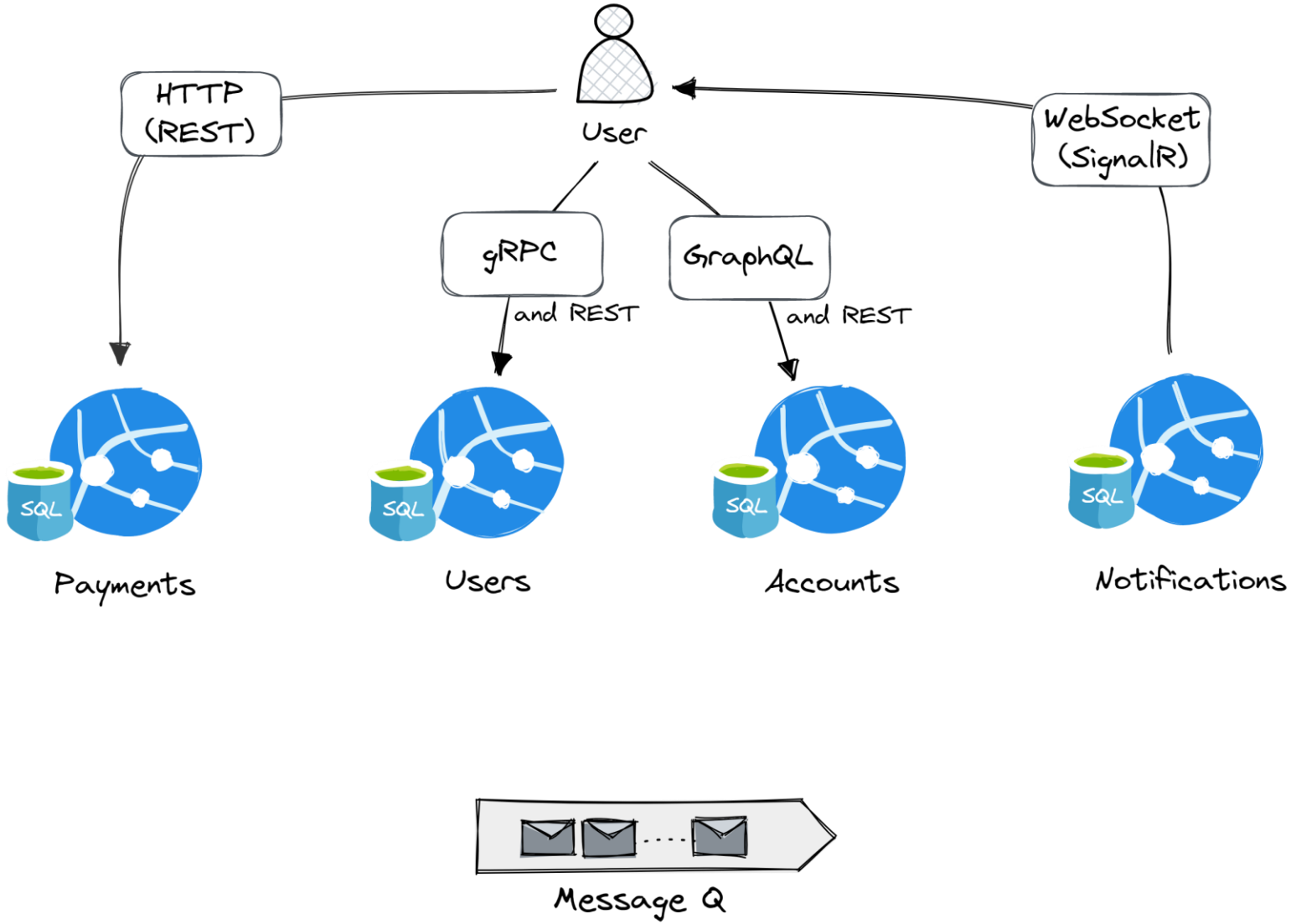
There’s a lot to unpack.



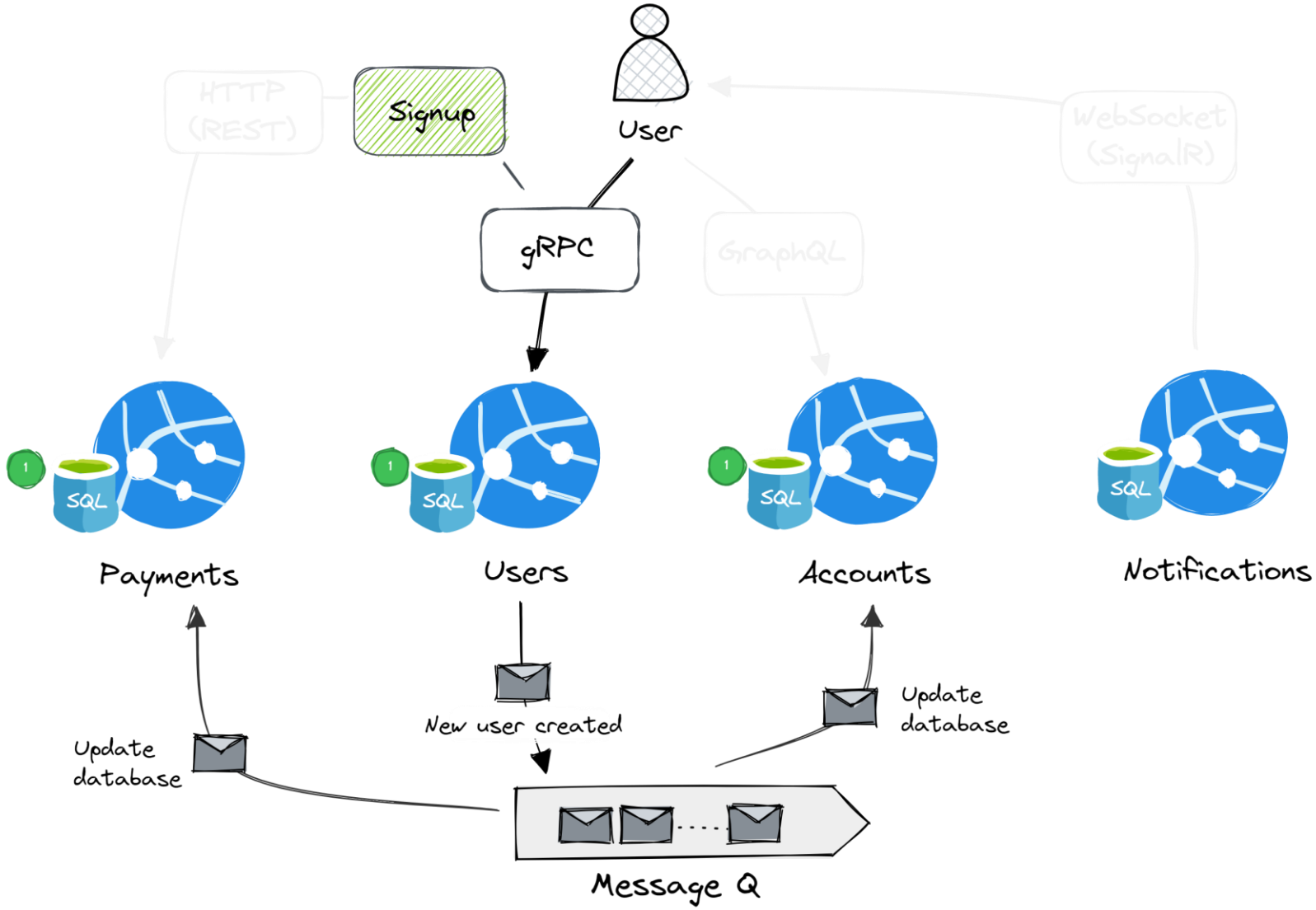
Our demo banking app solution architecture.



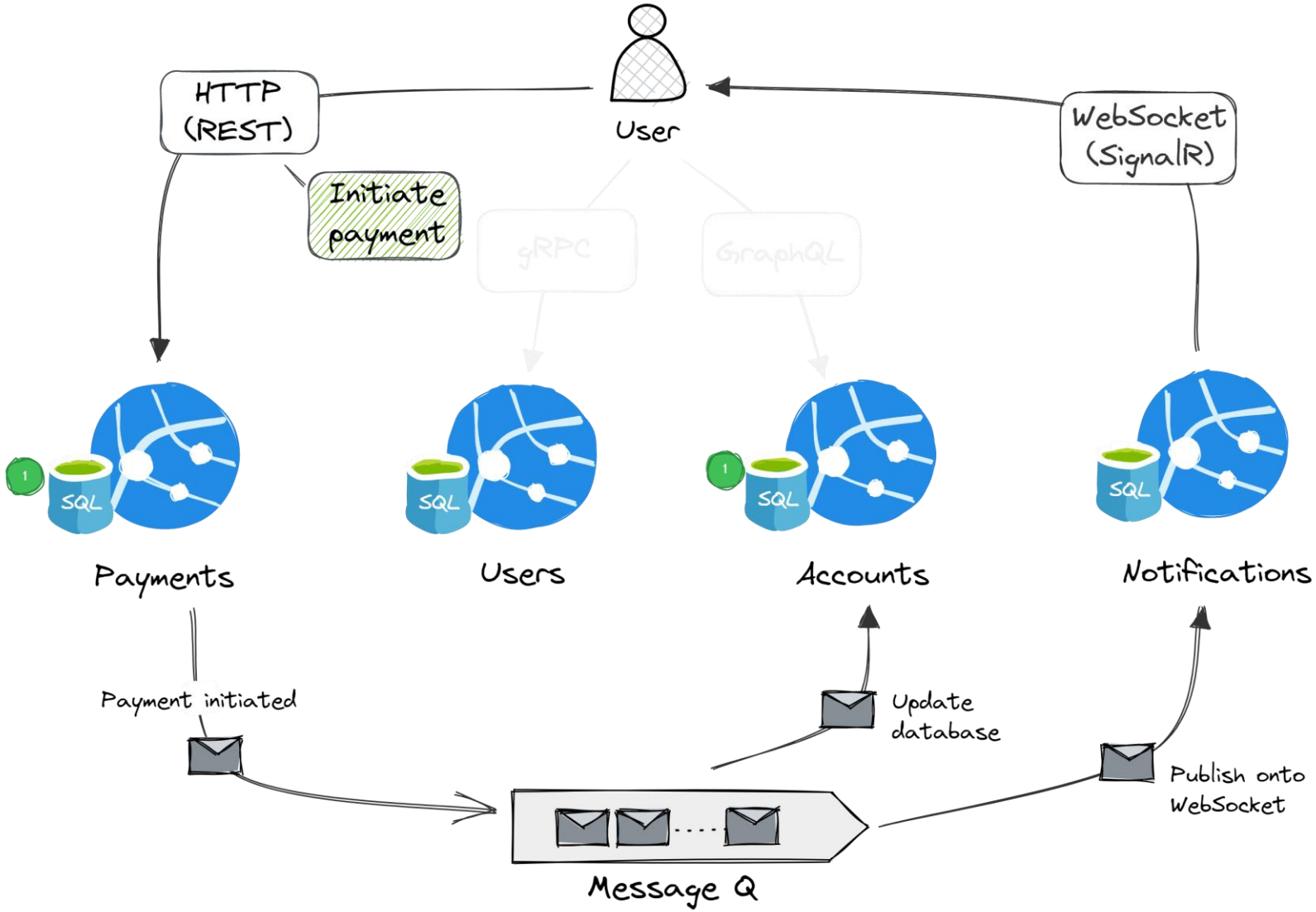
Our demo banking app solution architecture.



Sign-up using gRPC



Initiate a payment with HTTP

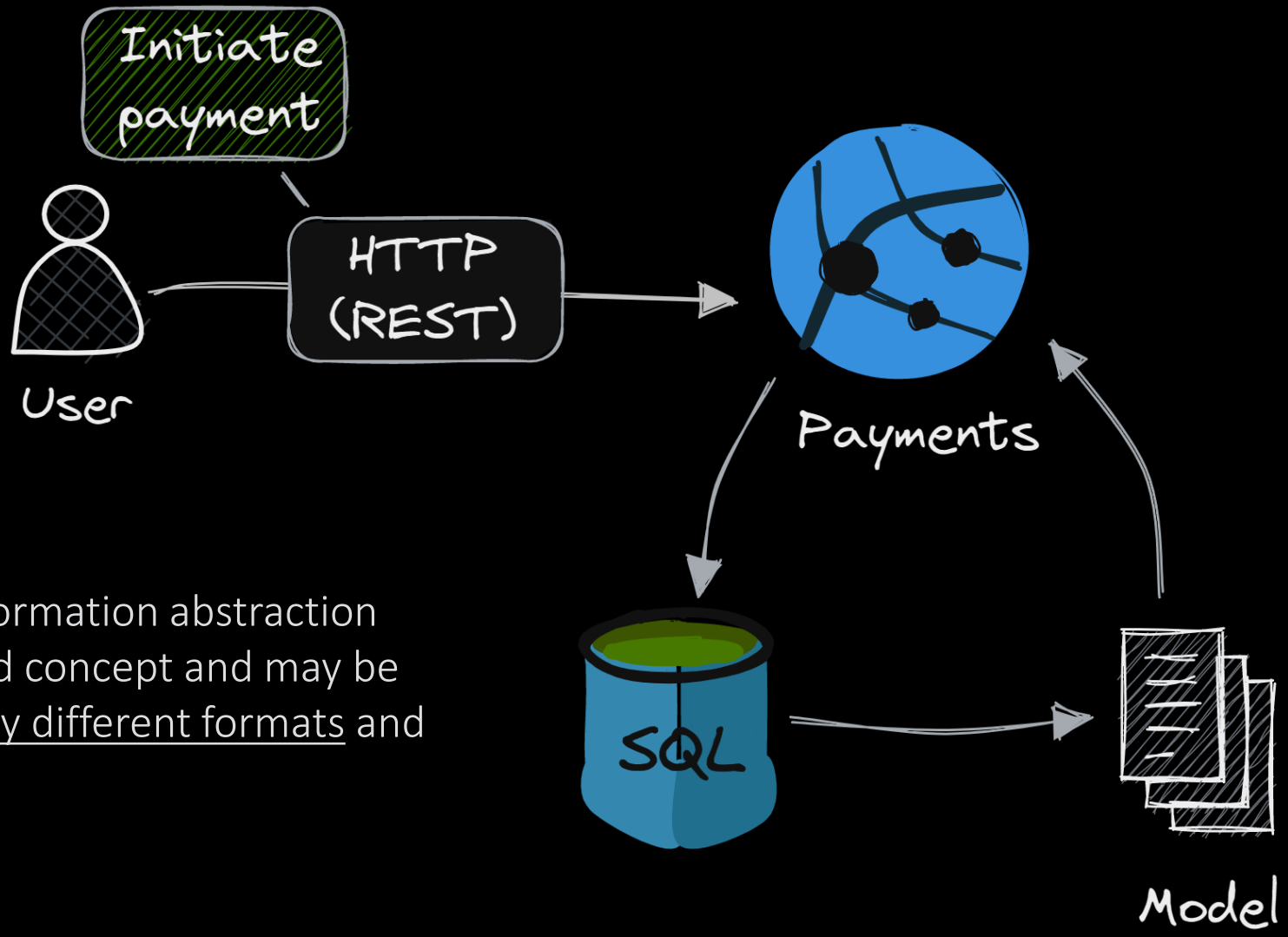


Why should you care?

REST*(ful). Everywhere.*

aka. JSON over HTTP **(... nowhere)**

How REST APIs are (typically) made.



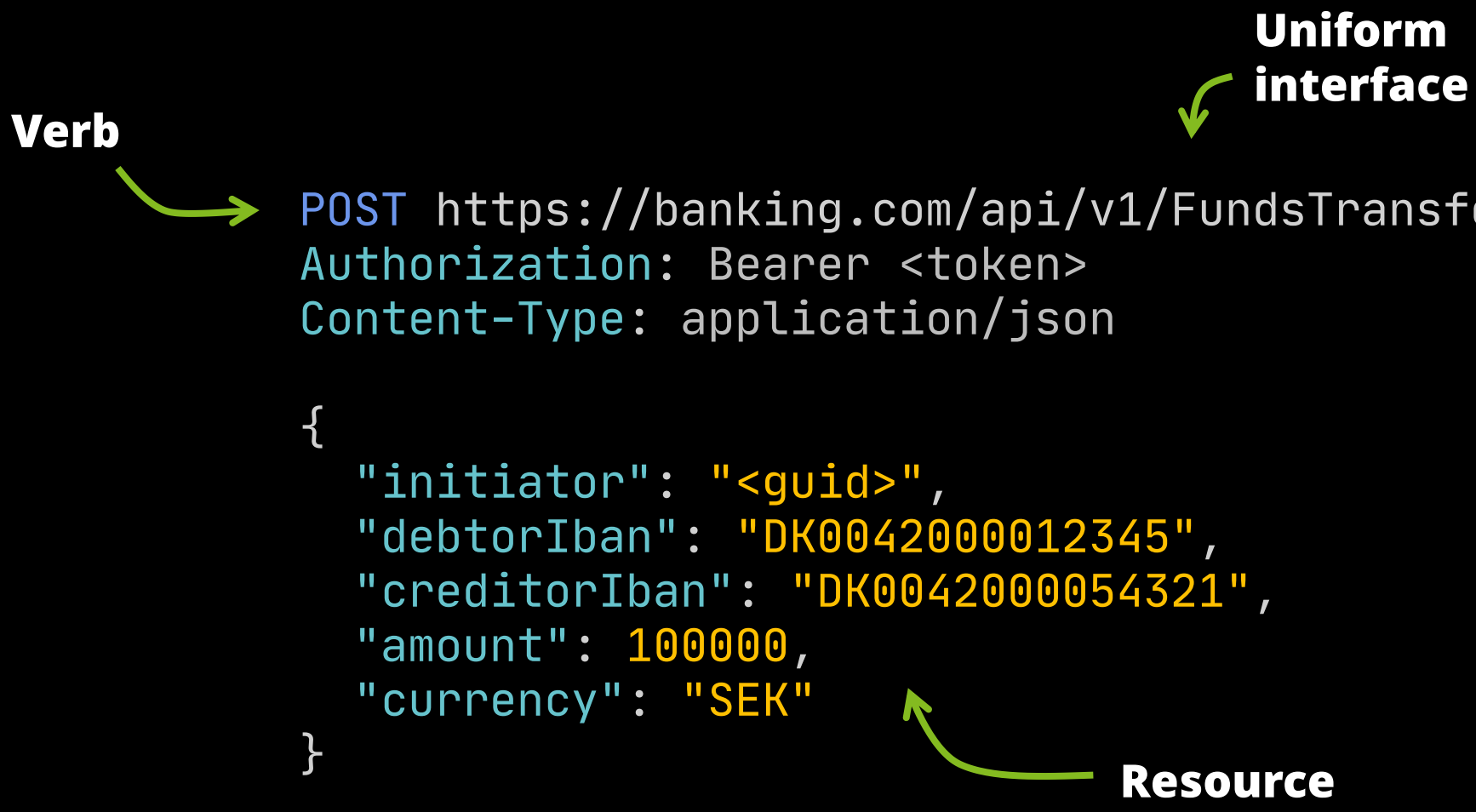
A resource is an information abstraction that covers a named concept and may be represented in many different formats and shapes.

Typical looking REST call.

```
POST https://banking.com/api/v1/FundsTransfer
Authorization: Bearer <token>
Content-Type: application/json

{
  "initiator": "<guid>",
  "debtorIban": "DK0042000012345",
  "creditorIban": "DK0042000054321",
  "amount": 100000,
  "currency": "SEK"
}
```

Typical looking REST call.



1-1 Anti pattern

```
{
  "initiator": "<guid>"
  "debtorIban": "DK0042000012345",
  "creditorIban": "DK0042000054321",
  "amount": 100000,
  "currency": "SEK"
}
```

Id	DebtorIban	CreditorIban	Amount	Currency	UserId
guid	DK4200213	DK0035131	10.000	DKK	guid

Glorified CRUD

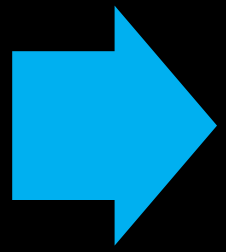
When a web developer marries a Database developer.

CREATE
POST

READ
GET

UPDATE
PUT

DELETE

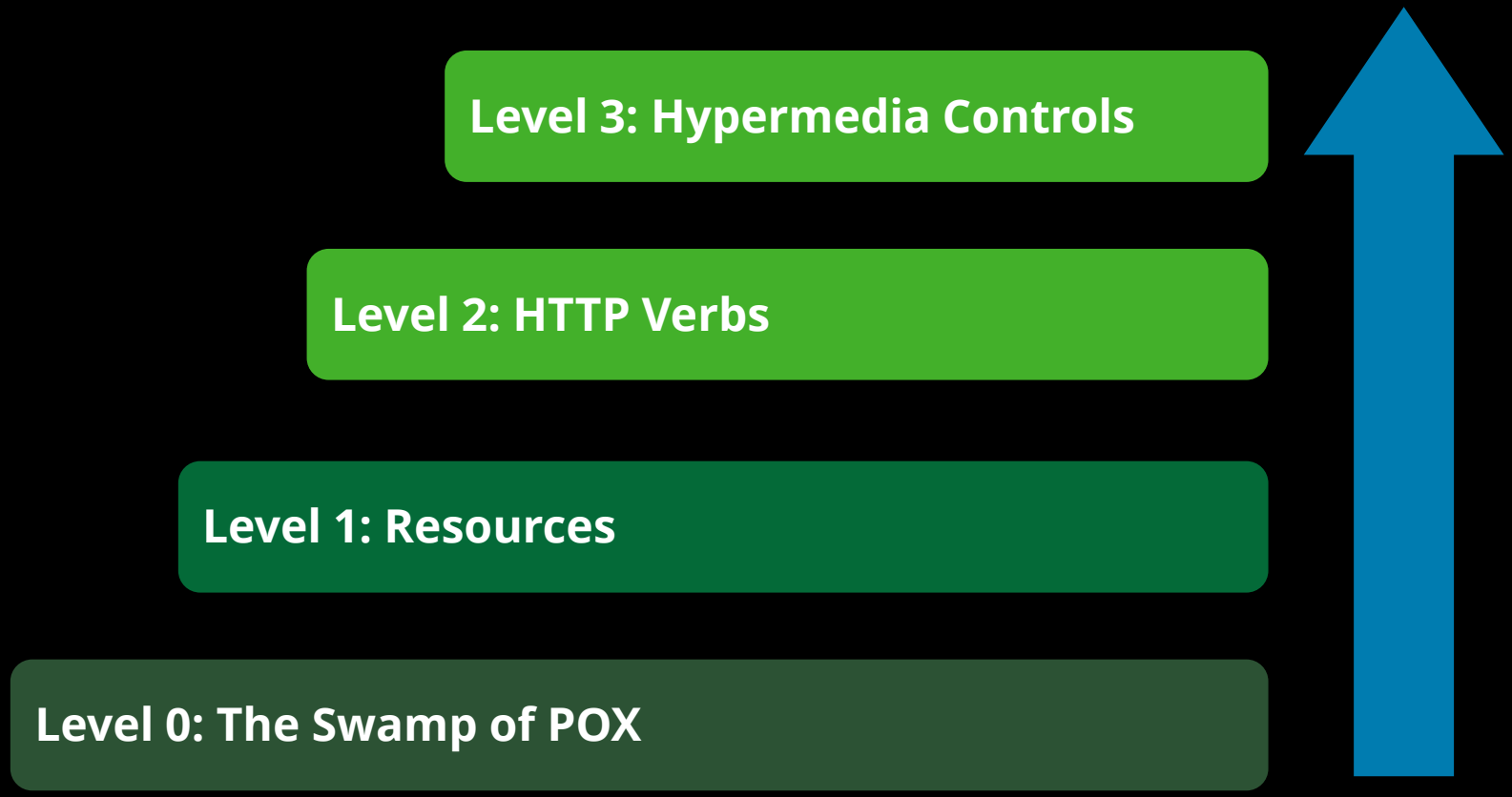


CRUD API

GET	api/v1/accounts
GET	api/v1/accounts/{id}
POST	api/v1/accounts
PUT	api/v1/accounts
DELETE	api/v1/accounts

Maturity model

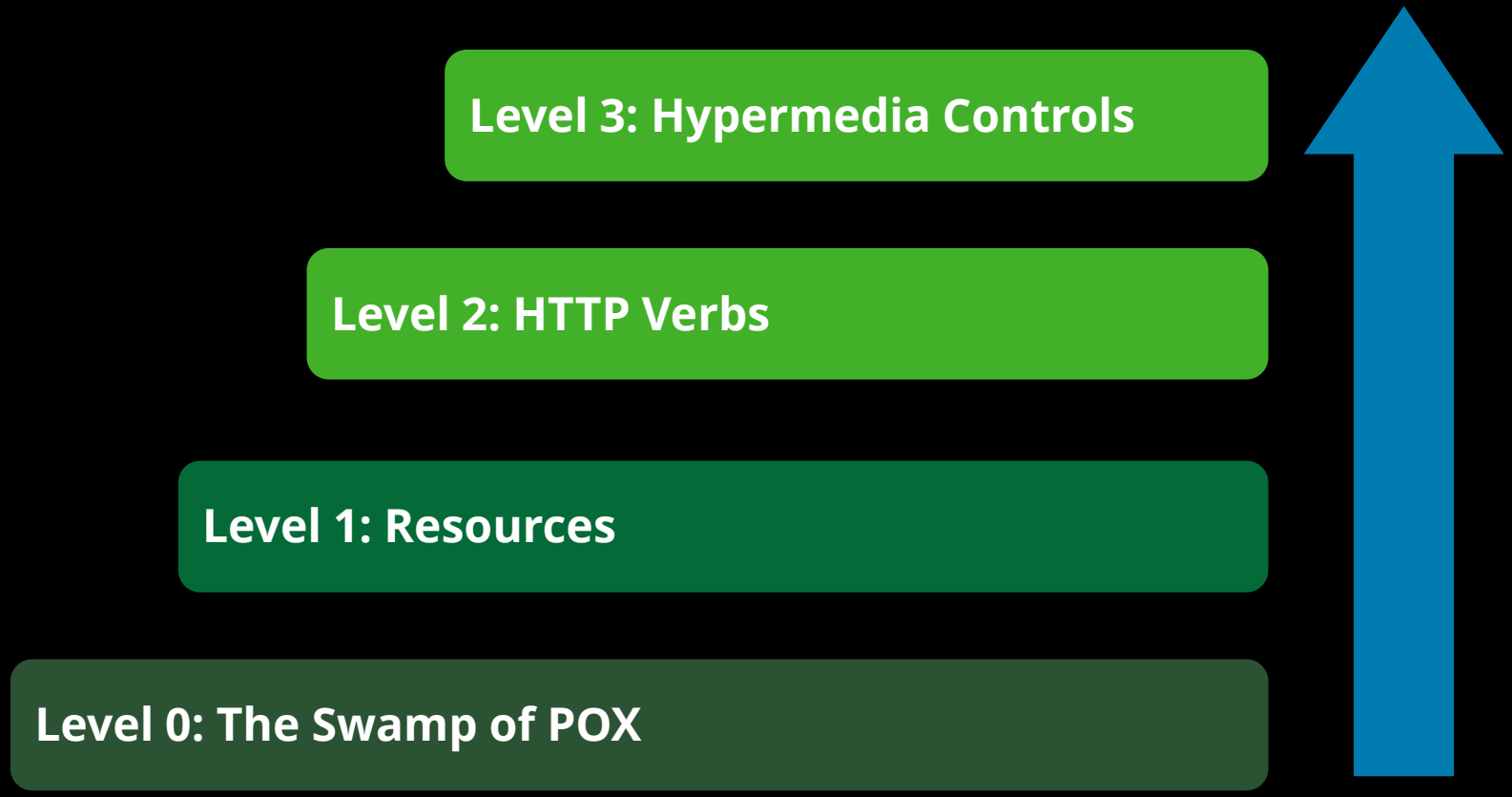
(Richardson REST maturity model)



Maturity model

(Richardson REST maturity model)

***This* is REST.**
Nothing else.



REST is hard to get right.
Today REST mostly means JSON over HTTP.

HATEOAS

Hypermedia as the Engine of Application State

```
POST https://banking.com/api/v1/FundsTransfer
Authorization: Bearer <token>
Content-Type: application/json

{
  "initiator": "<guid>",
  "debtorIban": "DK0042000012345",
  "creditorIban": "DK0042000054321",
  "amount": 100000,
  "currency": "SEK"
}
```

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
Location: https://banking.com/api/v1/FundsTransfer/transfer-id-guid
Transfer-Encoding: chunked

{
  "fundsTransferId": "transfer-id-guid",
}
```

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
Location: https://banking.com/api/v1/FundsTransfer/transfer-id-guid
Transfer-Encoding: chunked

{
  "fundsTransferId": "transfer-id-guid",
  "_links": [
    {
      ”name": ”Details",
      "href": "https://banking.com/api/v1/FundsTransfer/transfer-id-guid",
      "method": "GET",
      "description": "See funds transfer details"
    },
    {
      ”name": ”InitiatePayment",
      "href": "https://banking.com/api/v1/FundsTransfer",
      "method": "POST",
      "description": "Initiate a new transfer of funds"
    }
  ]
}
```

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
Location: https://banking.com/api/v1/FundsTransfer/transfer-id-guid
Transfer-Encoding: chunked

{
  "fundsTransferId": "transfer-id-guid",
  "_links": [
    {
      ”name": ”Details",
      "href": "https://banking.com/api/v1/FundsTransfer/transfer-id-guid",
      "method": "GET",
      "description": "See funds transfer details"
    },
    {
      ”name": ”InitiatePayment",
      "href": "https://banking.com/api/v1/FundsTransfer",
      "method": "POST",
      "description": "Initiate a new transfer of funds"
    }
  ]
}
```

“[…] if the engine of application state is not being driven by hypertext, then it cannot be RESTful and cannot be a REST API. Period.
– Roy T. Fielding

Versioning

Version 1?

```
GET https://banking.com/api/v1/accounts
```

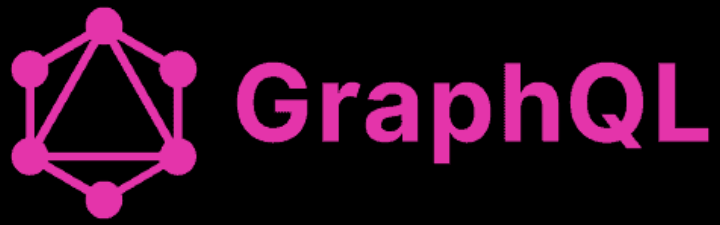
Version 2?

```
GET https://banking.com/api/v2/accounts
```

```
GET https://banking.com/api/accounts
Accept: application/vnd.accounts-complete+json
```

Use Content Negotiation

DEMO




```
query GetAllAccounts {
  allAccounts {
    iban,
    bookings {
      amount,
      currencyCode,
      debitCredit
    }
  }
}
```

POST

Query
Mutation

GET

Playground

The client dictates what's returned. Not the service.

```
### Get complete accounts representation

GET https://banking.com/api/Accounts
Accept: application/vnd.accounts-complete+json
```

```
query GetAllAccounts {
 allAccounts {
   userInfo: {
     id
     name: username # Rename
   },
   iban,
   bookings {
    amount,
    currencyCode,
    debitCredit
   }
 }
}
```

```
mutation openNewAccount {
  createAccount(account: {
    # Input
     currency: "DKK"
     iban: "DK4200874321"
     number: 874321
     ownerId: “<owner-id>"
  }) {
    # What we want returned
    iban
  }
}
```

Demo

GraphQL Playground

Is GraphQL an end to API versioning?

Not really.

Removing fields is still a breaking change.

Caching can be difficult.

Is it 1981 again?

```
syntax = "proto3";

option csharp_namespace = "Users.GrpcServer";

package usersgrpc;

service UsersService {
  rpc AuthenticateUser (AuthenticationRequest)
    returns (AuthenticationReply);

  rpc CreateUser (CreateUserRequest)
    returns (UserCreatedReply);

  rpc GetUser (GetUserRequest)
    returns (UserReply);
}

message AuthenticationRequest {
  string username = 1;
  string password = 2;
}

message AuthenticationReply {
  bool isAuthenticated = 1;
  optional string token = 2;
}

message CreateUserRequest {
  string username = 1;
  string password = 2;
}

message UserCreatedReply {
  oneof response {
    string userId = 1;
    CreationError error = 2;
  }
}

message CreationError {
  string message = 1;
}
```

Proto Description

```
syntax = "proto3";

option csharp_namespace = "Users.GrpcServer";

package usersgrpc;

service UsersService {
  rpc AuthenticateUser (AuthenticationRequest)
    returns (AuthenticationReply);

  rpc CreateUser (CreateUserRequest)
    returns (UserCreatedReply);

  rpc GetUser (GetUserRequest)
    returns (UserReply);
}

message AuthenticationRequest {
  string username = 1;
  string password = 2;
}

message AuthenticationReply {
  bool isAuthenticated = 1;
  optional string token = 2;
}

message CreateUserRequest {
```

Contract


syntax = "proto3";
option csharp_namespace = "Users.GrpcServer";
package usersgrpc;
service UsersService {
rpc AuthenticateUser (AuthenticationRequest)
returns (AuthenticationReply);
rpc CreateUser (CreateUserRequest)
returns (UserCreatedReply);
rpc GetUser (GetUserRequest)
returns (UserReply);
}
message AuthenticationRequest {
string username = 1;
string password = 2;
}
message AuthenticationReply {
bool isAuthenticated = 1;
optional string token = 2;
}
gRPC Server
gRPC Client
Stub
Stub
1
Protobuf Request
01101000 01100101 ...
2
Protobuf Response

Demo

Calling an gRPC method

Surprises.

Data modeling

Null handling - proto

```
message UserInfo {
  string userId = 1;
  optional string username = 2;
}

message UserReply {
  optional UserInfo user = 1;
}
```

Return value

```
return new UserReply
{
    User = new UserInfo
    {
        UserId = user.Id.ToString(),
        Username = null
    }
};
```

Tooling gotchas

(Hard to debug)

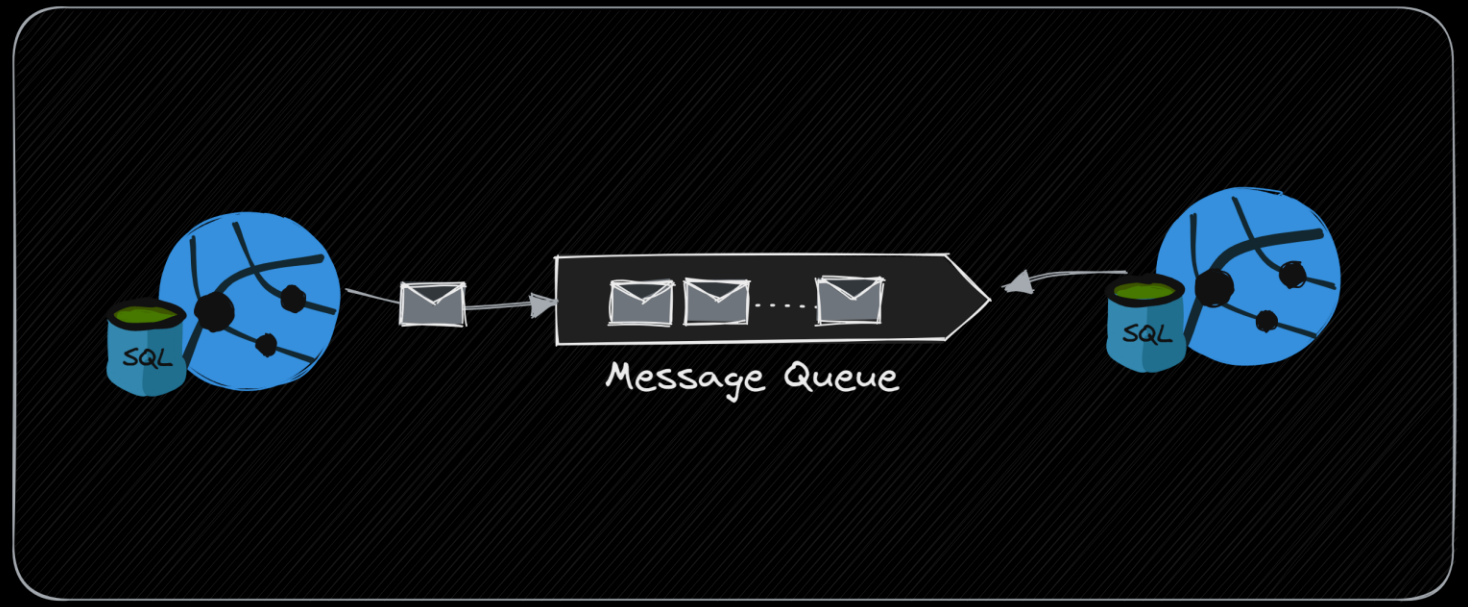
Out-of-sync
Proto descriptions

Messaging

Solution Architects’ dream.
Developers’ nightmare.

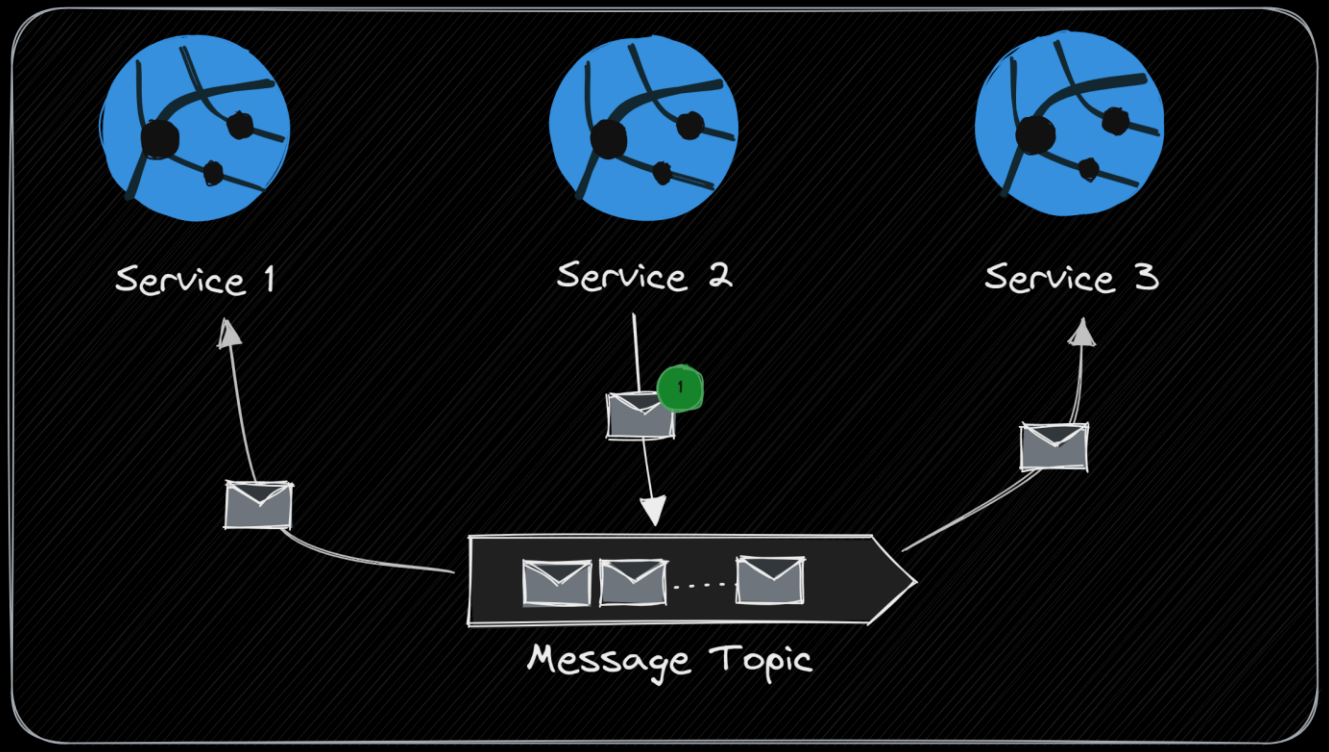
Core messaging concepts

Point-to-point



Core messaging concepts

Publisher/Subscriber



Typical event data

```
{
  "CorrelationId": "5ac3fe27-e3ee-47af-89b1-00c5765b38b1",
  "BeneficiaryIban": "DK4200987654",
  "OriginatingIban": "DK4200123456",
  "Amount": 100000,
  "Currency": "SEK"
}
```

Cloud Event

Our data

```
{
    "id": "78869fd3-a17c-4bb7-9a46-574be4564a01",
    "source": "payments-api/funds-initiated",
    "type": "payments.fundstransfer.initiated",
    "data": {
        "BeneficiaryIban": "DK4200987654",
        "OriginatingIban": "DK4200123456",
        "Amount": 100000,
        "Currency": "SEK"
    },
    "time": "2023-01-10T21:44:18.1689474+00:00",
    "specversion": "1.0"
}
```

cloudevents.io

Message formats

AMQP, MQTT, Avro?

AMQP

AMQP 0-9-1

Exchange **Binding** **Routing key**

Exchange type

Fanout
Direct
Topic
Headers

Queue

AMQP 1-0

Queue **Topic** **Subscription**

Demo

Publish and subscribing to messages

Lessons learnt

Use a shared library

(... but only to publish and subscribe)

Distributed message objects

Results in unintended, painful updates

Demo

Tying it all together.

Thank you!

Read my articles on medium nmillard.medium.com

Connect on LinkedIn linkedin.com/in/nicklasmillard

Reach out at nimillard@deloitte.dk