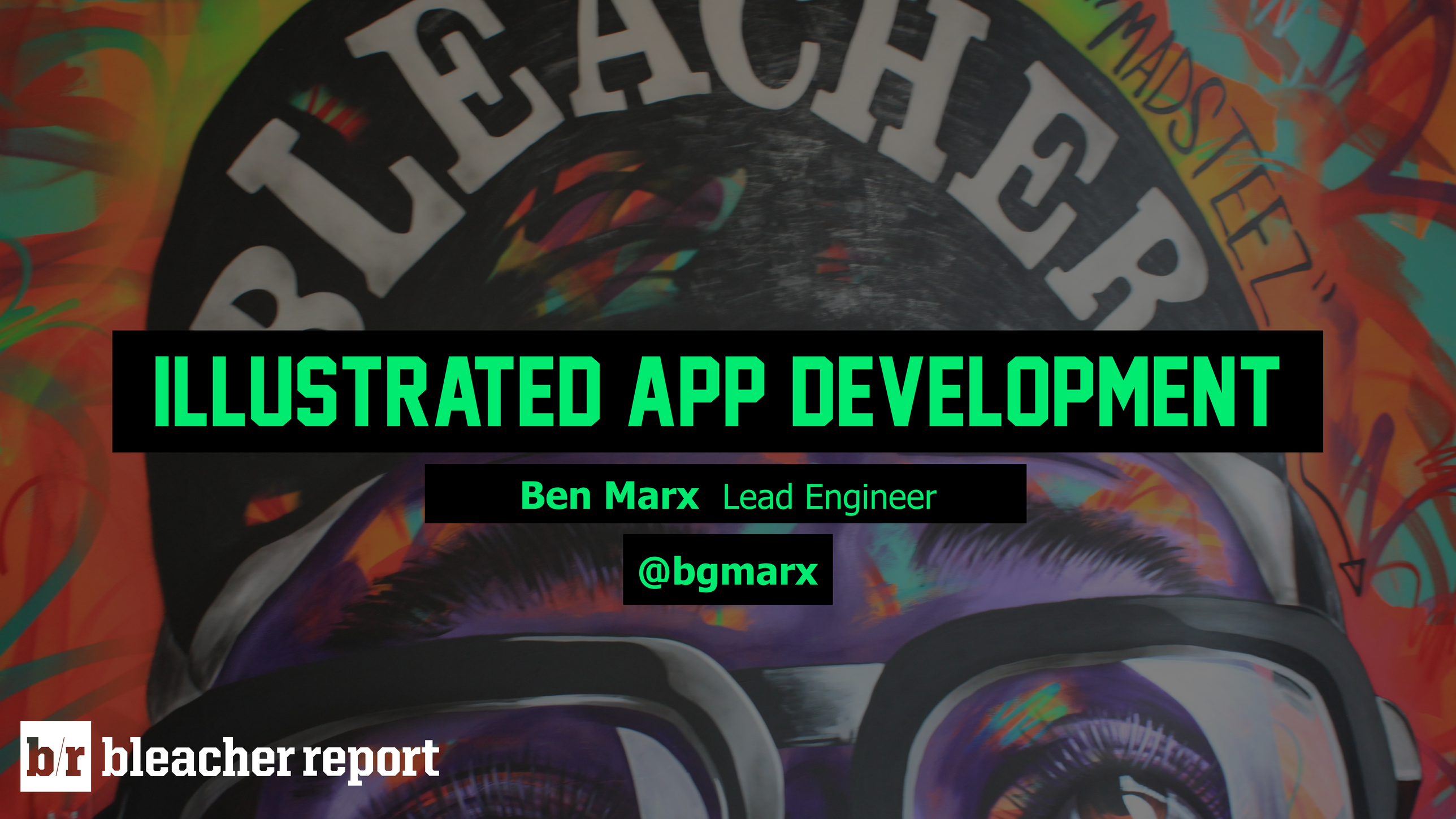


ILLUSTRATED APP DEVELOPMENT

Ben Marx Lead Engineer

@bgmarx

b/r bleacher report

STATS

2nd largest sports platform

- **15 million** app downloads
- **1.5 billion** global page views per month
- **16 billion** global page views in the last year
- **250,000 concurrent** users at peak
  - The Lebron “Decision”
  - NFL Draft
  - Unexpected breaking news
- **Over 3 billion** push notifications per month

TECHNICAL DEBT

Reasons for exploring alternatives to Ruby and Rails

- **8 years**
  - Started Rails 1.x
  - Monolith
  - Web first
  - Lost domain knowledge
  - Shifting trends

INITIAL CONSIDERATIONS

Business requirements to meet when choosing a new language

- **Breaking** News, Notifications
  - Be the **fastest**
  - Not Ruby's strength
    - **Caching**, server costs
- **Customizable**
  - On-the-fly content delivery
  - **No or limited caching**
  - Fantasy players

WHY ELIXIR

From a number of options, Elixir was chosen

- **New** language
  - Driven by respected developers and contributors
  - Except for Erlang, the BEAM
    - Multi-core world
  - Can always fall back to Erlang if necessary
- **Familiar** Syntax
  - Ruby is comfortable
  - Erlang is unique

APP LIFECYCLE

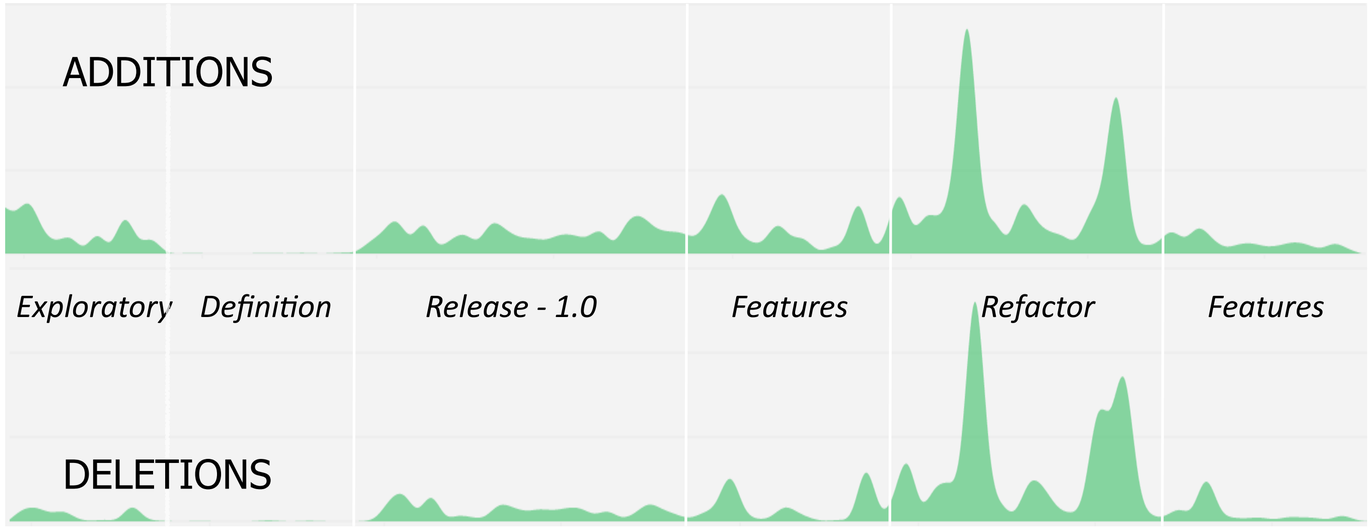
APP LIFECYCLE

The common lifecycle patterns of most applications

- Product Requirements
- Initial Development
- Features
- Release
- Refactor
- Features

APP LIFECYCLE

The common lifecycle patterns of most applications



EXPLORATORY

EXPLORATORY

- **Initial Commit October 2014**
- **Rails-y**
- **Simple data model**
  - Stream has many items
- **Deploy**
  - Docker
  - Elastic Beanstalk

Documentation

API documentation is available at http://api.phoenixframework.org/

Development

There are no guidelines yet. Do what feels natural. Submit a bug, join a discussion, open a pull request.

Building phoenix.coffee

```
$ coffee -o priv/static/js -cw priv/src/static/cs
```

PRODUCT DEFINITION

PRODUCT DEFINITION

- **Initial success**
- **Developer approval**
- **Second Elixir app**
- **Monolith**
  - Lost domain knowledge
- **Ecto Changesets**

```
def update_item_from_map(item_map) do
  item = from_map(item_map)

  Repo.update(item)
  item
end
```

RELEASE

RELEASE

- **Stagnant**
  - Versions behind Phoenix
  - Churn and dramatic change
  - Expected from pre-1.0
- **Lessons learned**
  - Tech debt at bay
  - Refactor as integral part of development

RELEASE

- **Migrated all streams to new app**
  - Phased rollout
  - Least popular to most popular
  - Rollout Flags
- **Performant App**

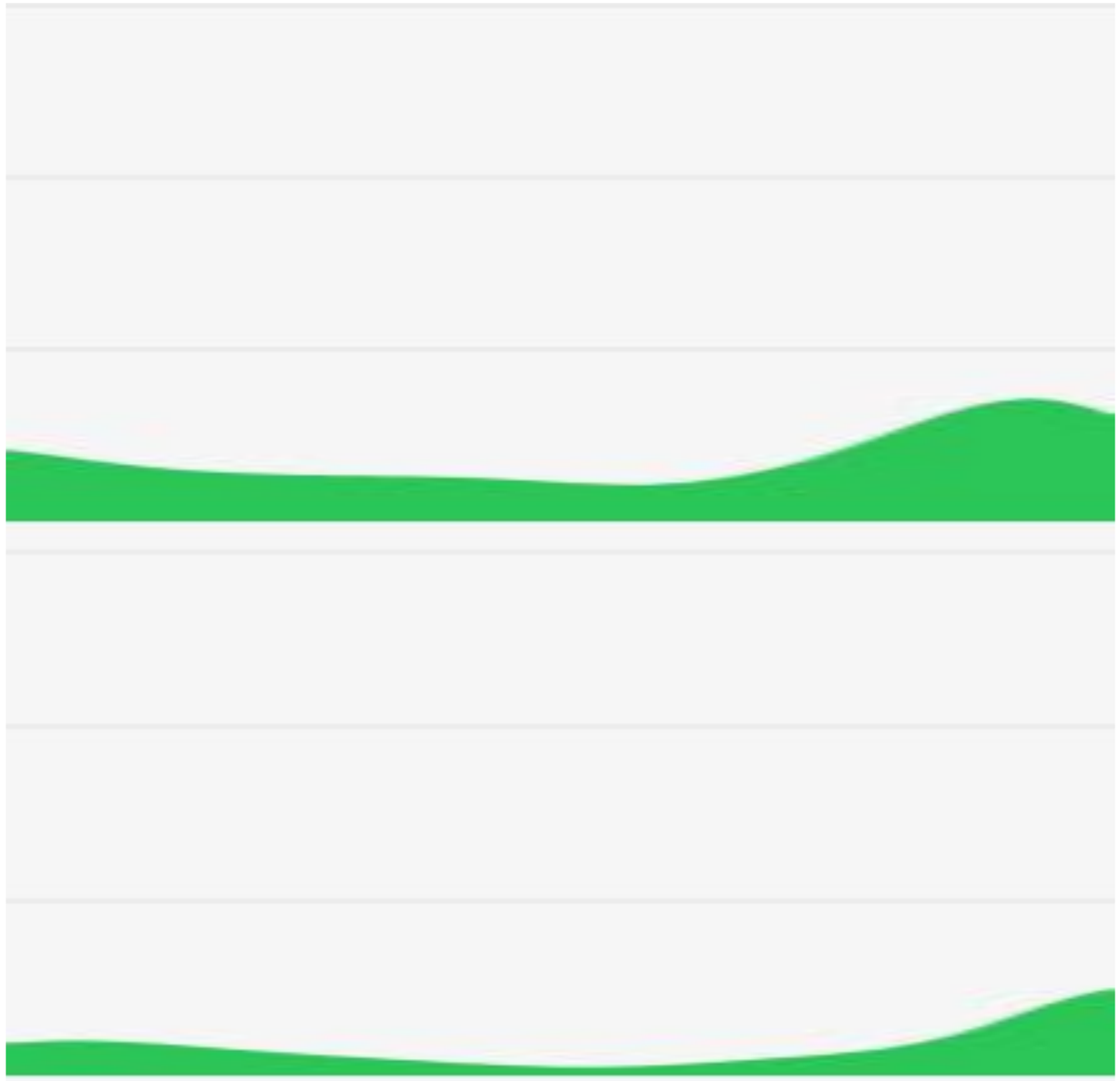
RELEASE

- **Still Object Oriented**
  - Model and interactors
- **Still lacking library support**
  - New Relic
    - Exometer

FEATURES

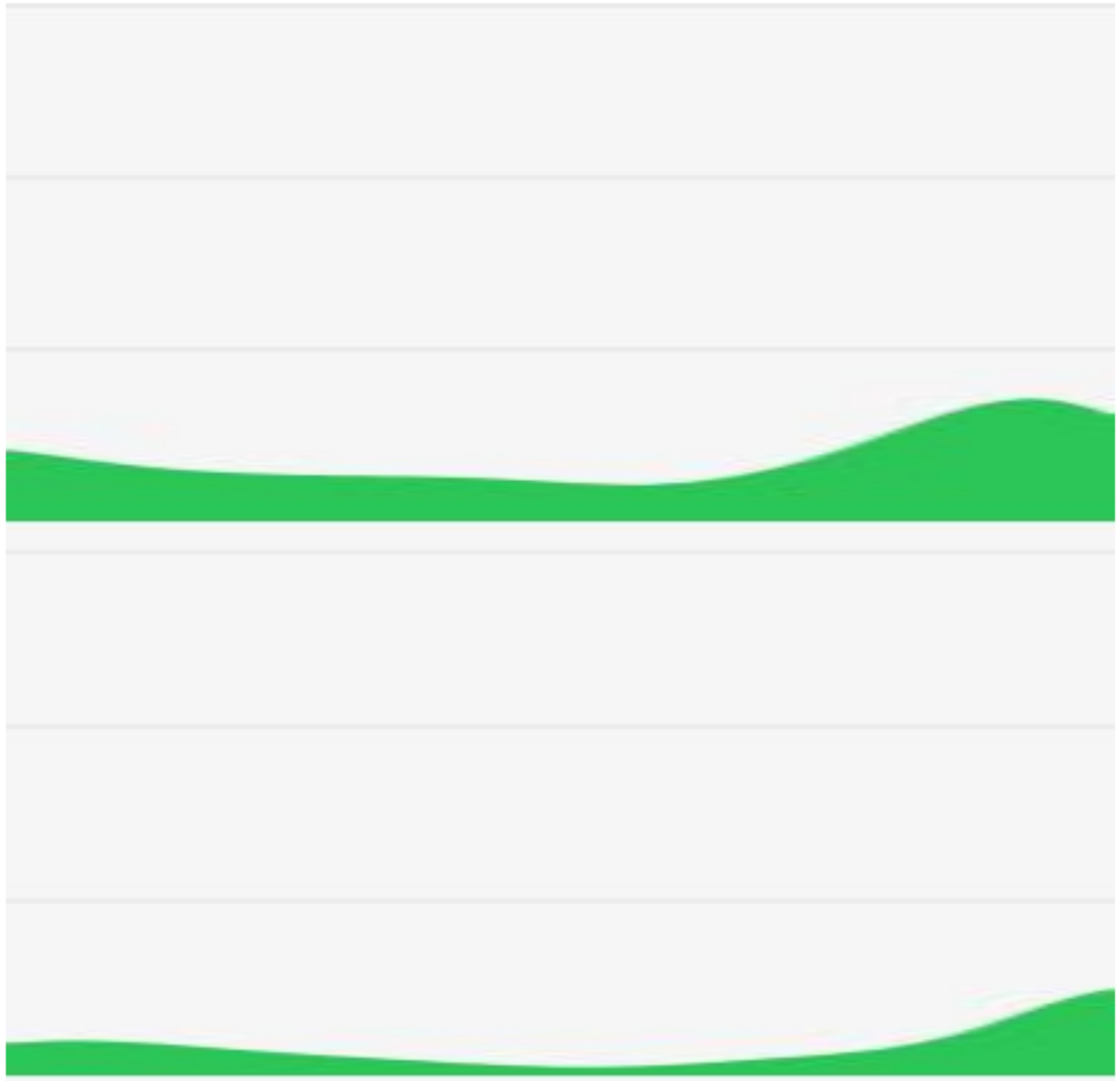
FEATURES

- 3 Elixir apps
- No major releases



FEATURES

- **Feature development on hold**
  - Big backlog
- **New content types**
  - Difficult to validate
  - Non-trivial work



REFACTOR

REFACTOR

- **Huge undertaking**
  - Many moving parts
- **Informed data modeling**
  - Stream has many items
    - has one content
      - has one metadata

REFACTOR

- **Trivial to add content types**
  - Ecto Changesets
  - Good way for new developers to start working with Elixir
  - Easy to test

```
def insert_changeset(model, params \\ :empty, content_type) do
  model
  |> cast_content_type(params, content_type)
  |> cast(params, ~w(), @optional_fields)
end

def cast_changeset("video_article", changeset, params) do
  changeset
  |> cast(params, @required_article_fields, @optional_fields)
  |> set_video_autoplay(params["autoplay"])
  |> validate_inclusion(:hook_type, @hook_types)
  |> validate_gif_hook
end
```

REFACTOR

- **Standardizations**
  - Credo
  - Excoveralls
  - Inch
- **Consistent code style**
- **More meaningful code reviews**

```
Refactoring opportunities

[R] ↗ If/else blocks should not have a negated condition in `if`.
      lib/phoenix/channel.ex:26 (Phoenix.Channel.subscribe)
[R] → Function is too complex (max ABC is 15, was 43).
      lib/phoenix/router.ex:563:8 (Phoenix.Router.add_resources)
[R] → Function is too complex (max ABC is 15, was 16).
      lib/phoenix/router/socket.ex:12:12 (Phoenix.Router.Socket.channel)
```

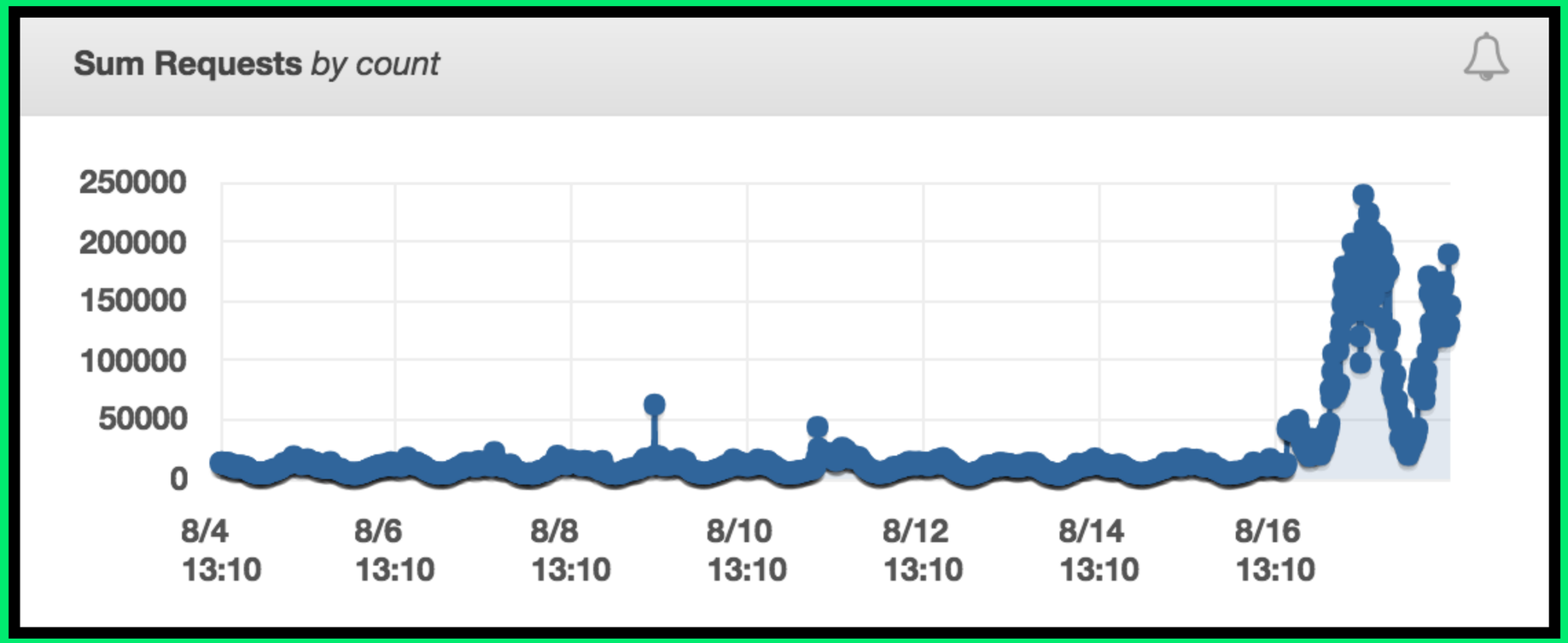
FEATURES

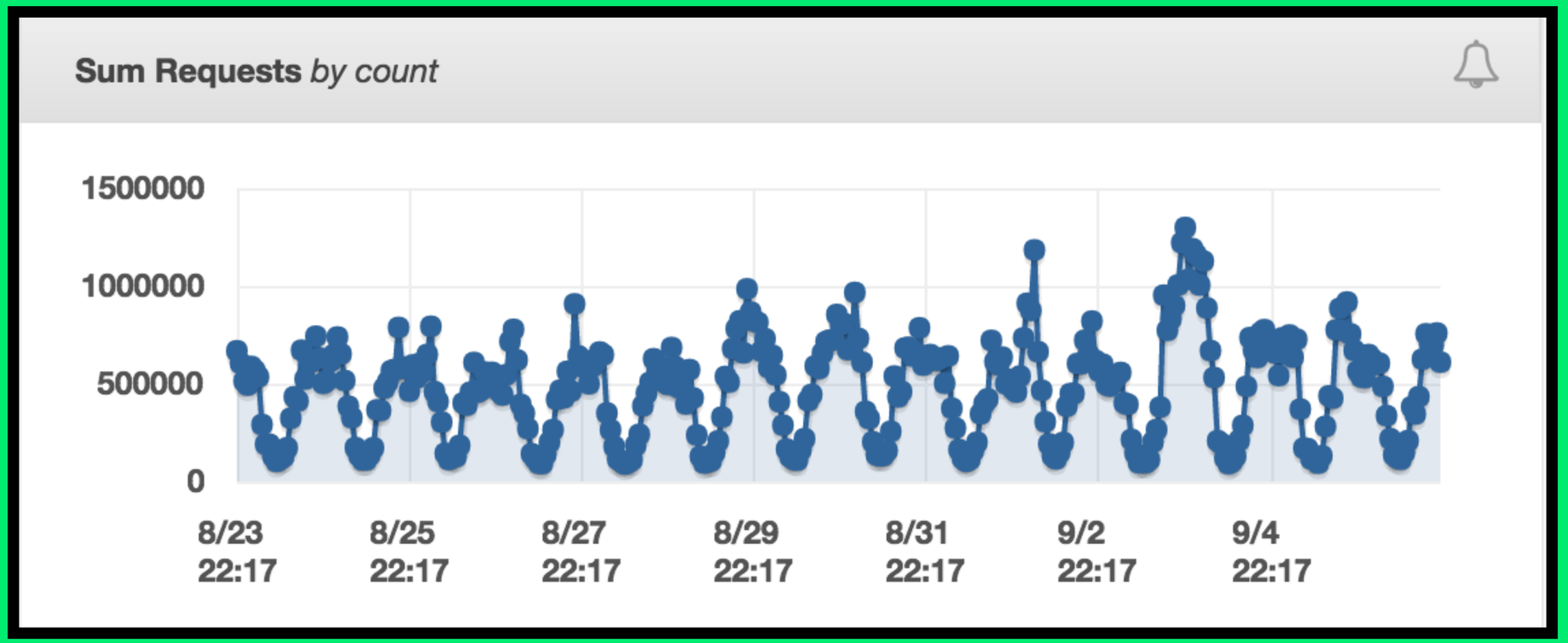
FEATURES

- **4 Elixir apps**
  - 2 in development
- **NFL draft**
  - Barometer for company
  - First major test
- **Kevin Durant Free Agency**
  - Traffic increased 4x in minutes

FEATURES

- **Move all list blending to one app**
  - Most hit endpoint in the system
- **Understanding the monolith better**
  - Better architectural decisions
- **Leverage Fastly**


Sum Requests by count
250000
200000
150000
100000
50000
0
8/4
13:10
8/6
13:10
8/8
13:10
8/10
13:10
8/12
13:10
8/14
13:10
8/16
13:10


Sum Requests by count
1500000
1000000
500000
0
8/23
22:17
8/25
22:17
8/27
22:17
8/29
22:17
8/31
22:17
9/2
22:17
9/4
22:17

ENCOURAGE ITERATION

ECTO CHANGESETS

```
def update_track_from_map(item_map) do
  item = from_map(item_map)

  Repo.update(item)
  item
end
```

```elixir
def changeset(model, params \\ :empty) do
  model
  |> cast(params, @required_fields, @optional_fields)
end
```

```
Wrapped in a transaction:

def update_changeset(data, params \\ :invalid) do
  data
  |> cast(params, @required_fields, @optional_fields)
  |> validate_inclusion(:content_type, @content_types)
  |> validate_url(params["content_type"])
  |> standardize_url
  |> lock_position
  |> update_position
  |> unique_constraint(:position, name: :items_unique_position)
  |> unique_constraint(:url, name: :items_unique_url)
  |> update_url
end
```

```
Parent:
def update_changeset(data, params \\ %{}) do
  data
  |> cast(params, @fields)
  |> validate_required(@required)
  |> validate_length(:url, min: 4)
  |> validate_inclusion(:content_type, @content_types)
  |> validate_url(params["content_type"])
  |> unique_constraint(:position, name: :items_unique_position)
  |> unique_constraint(:url, name: :items_unique_url)
  |> status
  |> cast_assoc(:content, required: true, with: &App.Content.update_changeset(&1, &2,
params["content_type"]))
end

Child:
def update_changeset(data, params \\ %{}, content_type) do
  data
  |> cast(params, @fields)
  |> cast_content_type(params, content_type)
  |> cast_assoc(:metadata, required: true, with: &App.AssociatedData.update_changeset(&1, &2,
content_type))
end
```

BULK ACTIONS

```
streams
|> Enum.map(&Task.async(__MODULE__, :create, [item_hashes, &1]))
|> Enum.map(&Task.await(&1)
|> List.flatten
```

```
Stream.filter(item_hashes, &(&1["url"] != nil))
|> Enum.uniq(&(&1["url"]))
|> Stream.map(&Task.async(Track, :fetch_embed,[&1, skip_embed]))
|> Enum.map(&Task.await(&1, 30_000))
|> Enum.map(&hash_to_track/1)
|> Enum.filter(&(&1.embed["errors"] == nil ))
|> Enum.map(&create_track(&1, Stream.find_or_create(stream)))
```

```
Enum.map(Enum.chunk(streams, chunk_size, chunk_size, []), (
      fn chunk ->
        Task.async(
          fn  ->
            Enum.map(chunk,
              fn stream ->
                create_tracks(stream, item_params)
              end)
          end)
      end))
      |> Enum.map(&Task.await(&1))
      |> List.flatten
```

ELIXIR AT BR

ELIXIR AT BR

So where do we go from here?

- **6 Elixir apps**
  - 3 more in development
  - New consumer facing apps are Elixir
- **Happy stakeholders**
  - Tangible benefits
    - Server costs
    - Third party integrations

ELIXIR AT BR

So where do we go from here?

- **All backend developers write Elixir**
  - Some frontend, too
  - **Happy** and **productive**
  - Quickly writing production ready code
  - Open source
  - Dev blog @ dev.bleacherreport.com

QUESTIONS

Ben Marx
@bgmarx
bmarx@bleacherreport.com

b/r bleacher report