Turbocharging React Native Performance ⚡

Field Notes from Optimising an E-Commerce App

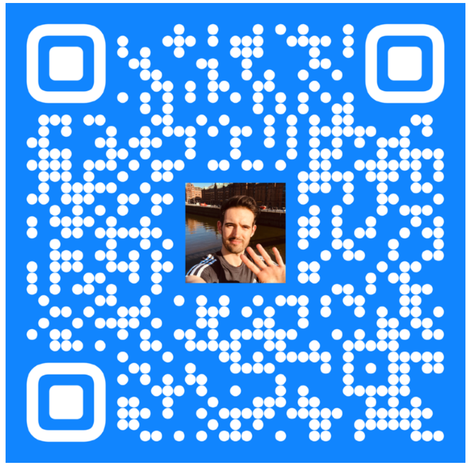
Jonathan Bones • 18.06.2025

Hi 👋

- 🚀 Meetup co-organizer
- 📱Passionate about mobile
- ⚛️ 7+ years of experience with React Native

Jonathan Bones

@bonesyblue



bitglow

bitglow

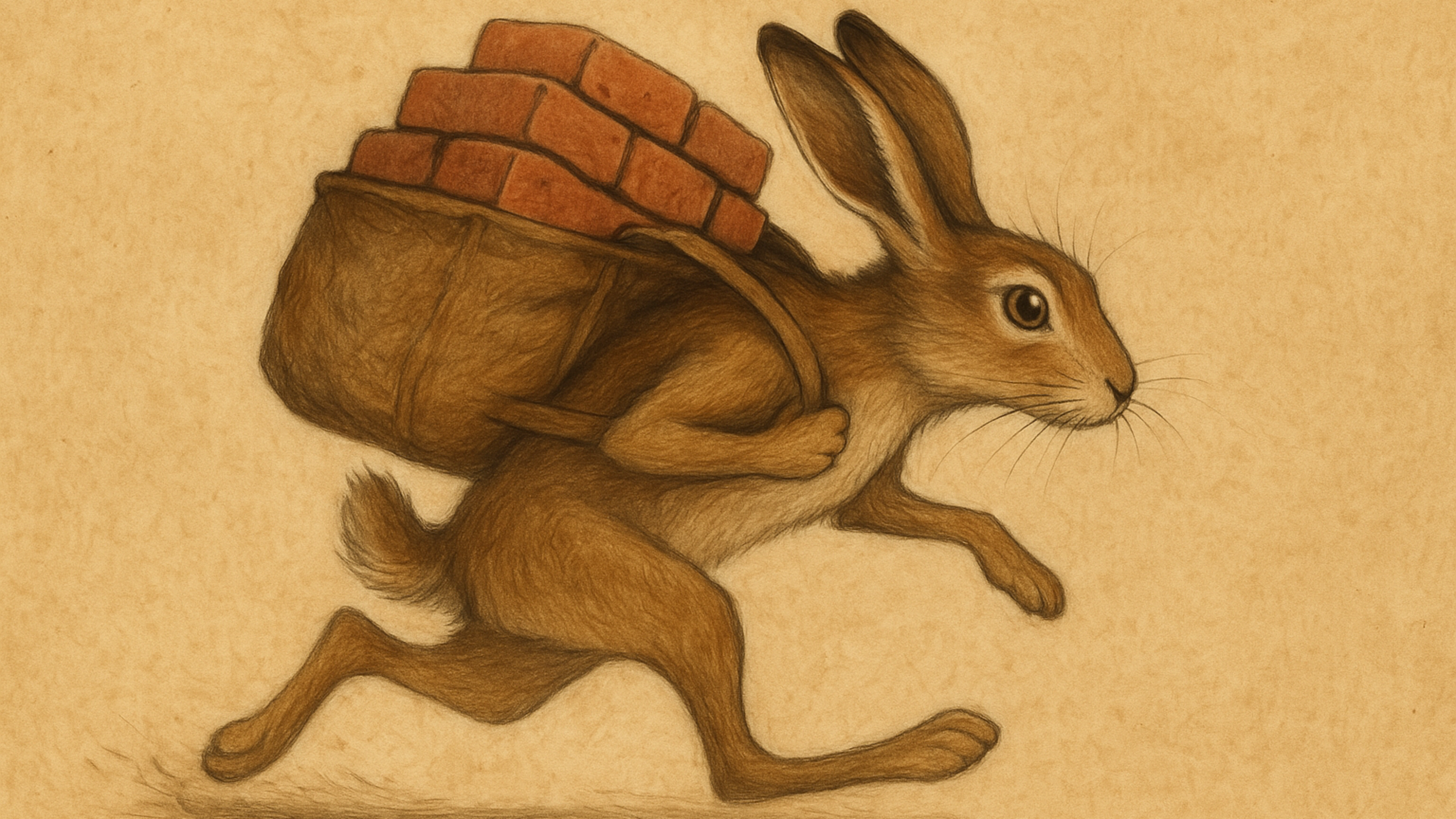
- ⚓ React & React Native specialists based in Kiel
- 🌍 Cross-industry experience with projects in the healthcare, financial services, e-commerce, budget planning and weather services
- 👨‍💻 Comprehensive delivery support - assisting at every stage from ideation and feature planning to development, CI/CD, app store management and release
- 🎯 Commitment to delivering quality

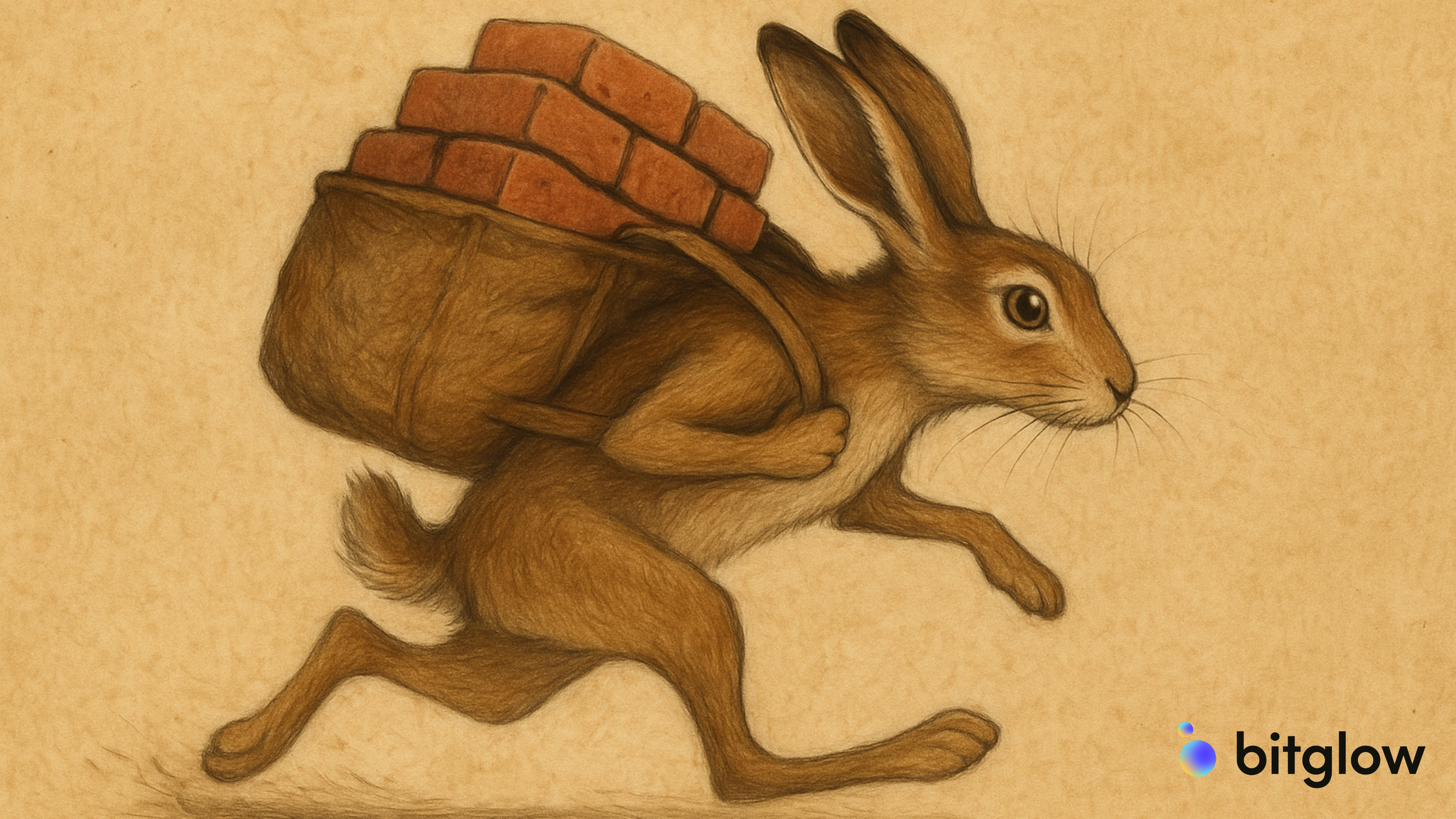
Turbocharging React Native Performance ⚡

Agenda

- 🔍 Gathering key metrics
- 🍾 Identifying performance bottlenecks
- 🐾 Designing & implementing a multi-phase improvement plan
- 📈 Incrementing on Performance

bitglow





bitglow

What The Users Think ⭐

★☆☆☆☆

Translated from German - Show original review

The app could be so great if it didn't keep crashing. You constantly have to wait because everything takes forever to load or doesn't load at al l. Apparently, I often don't have an internet connection. It's strange that I can then access the internet without any problems in the browser or other apps.

★☆☆☆☆

Translated from German - Show original review

It's worse than Autobahn Police Simulator 2. -_-

★☆☆☆☆

Translated from German - Show original review

I've rarely seen such a bad app. When you try to place an order, the address entry is suddenly interrupted with an error message saying the sh opping cart is empty. But it wasn't... The mug I had in my cart along with three other items was only available once, and now I can't order it. W hat a shame! Time to switch to other shops and apps!

bitglow

💬 What The Users Are Really Saying

“...You constantly have to wait because everything takes forever to load...”

⏳ **A long time to interactive (TTI) is negatively impacting the user experience**

“...Time to switch to other shops and apps...”

🔀 **User frustration is losing sales and leading to customer churn**

“...It's worse than Autobahn Police Simulator 2...”

🤔 **Yeah not sure what the takeaway is here either...**

bitglow

App Performance Isn’t Just Technical - It’s Commercial

💡 Takeaway no. 1

bitglow

🔍 Gathering Key Metrics

bitglow

Deciding on Metrics

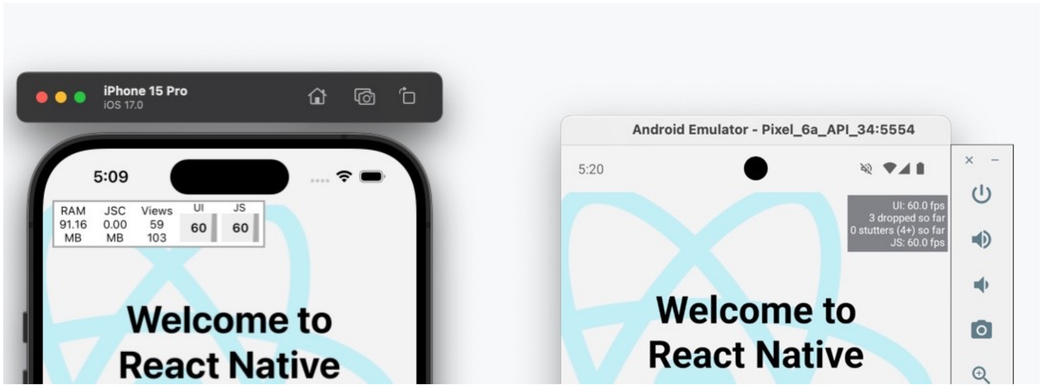
- Agree on a baseline for comparison
- What to measure? 🤔

bitglow

Deciding on Metrics

🚀 RN Performance Monitor

- Available via RN dev menu
- Provides Insight into JS & Native Thread frame rate
- Frame rate impacts perceived smoothness
  - Smooth apps run at 60 FPS
- Runtime only



bitglow

demo

React Native Performance Monitor

bitglow

Deciding on Metrics

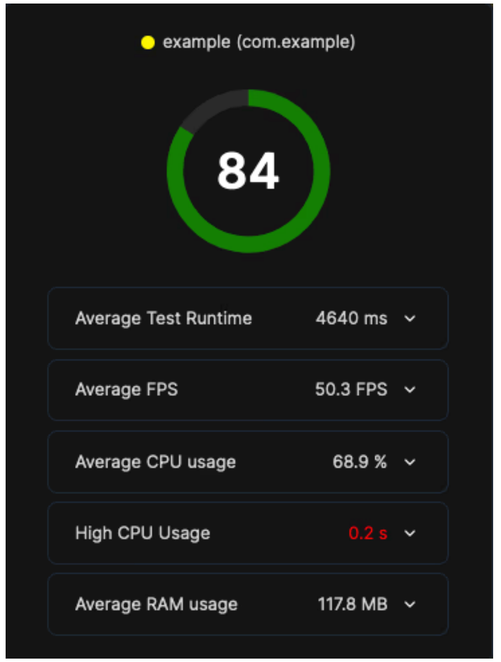
🔦 Lighthouse CLI

- Lighthouse-style performance score

$$Score = \frac{(UIfps + JSfps) \times 100}{120} \times (1 - JSthreadlock\ percentage)$$

Alexandre Moureaux (2022). Available at: https://apps.theodo.com/en/article/measuring-and-improving-performance-on-a-react-native-app

- Automate measurements with maestro CLI



bitglow

demo

Lighthouse + maestro CLI 💖

bitglow

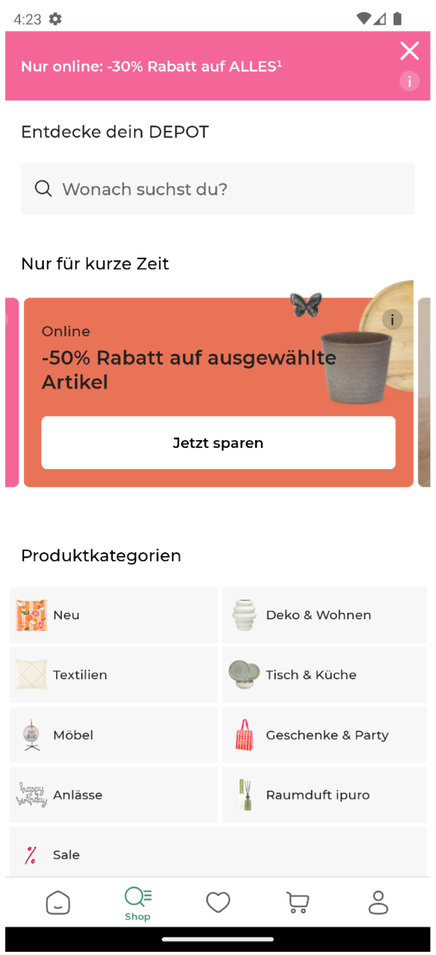
Deciding on Metrics

- Agree on a baseline for comparison
- What to measure? 🤔
  - Flashlight score for critical user path
  - Target: Flashlight score of **85** 🎯

bitglow

DEPOT

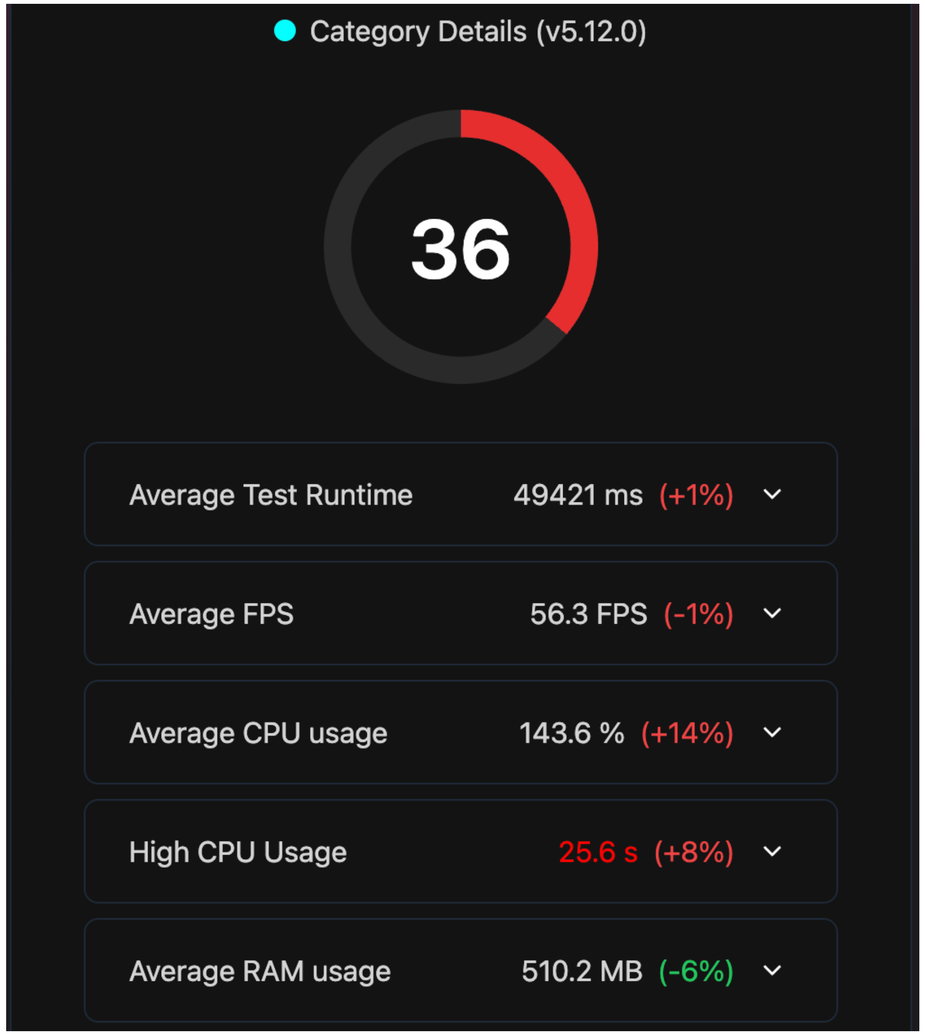
- Interior decorations retailer with a chain of shops in DE
- Omni-channel online store with web & mobile app
- Trusted partner since 2021
- Cross-platform codebase for iOS & Android
- Powered by Expo 💙



bitglow

Status Quo 😬

- Low flashlight score: **36**
- JS thread is blocked significantly
- Average FPS is close to 60 FPS target (~94%)
- High CPU usage ~50% of test run
- One CPU core is blocked on average for the entire test run
- Laggy UX & battery drain 🪫



bitglow

Identifying performance bottlenecks

bitglow

Common Pitfalls 🚩

- Image size 🖼️
- Virtualization 👻
- Networking 🕸️
- Too many re-renders ♻️

bitglow

Image Size 🖼️

Pitfall: Rendering uncompressed or oversized images 🚩

- Longer to download
- Memory pressure from decoding & resizing

Flashlight Impact 🔦

- FPS → Drops when decoding / resizing on main thread
- CPU usage → spike during decoding / resizing
- Increases page load time & TTI

bitglow

Resize images on the server

💡 Takeaway no. 2

bitglow

Virtualization 👻

Pitfall: No virtualization of lists / tables 🚩

- Views rendered at once
- Memory spike on first mount

Flashlight Impact 🔦

- FPS → Drops heavily when mounting
- CPU usage → spike on first mount
- Higher thread lock and laggy UI

bitglow

Only render what's necessary or soon to be visible

💡 Takeaway no. 3

bitglow

Virtualization 👻

bitglow

Networking 🕸️

Pitfall: Large number of small network requests (e.g. per-list item) 🚩

- Increased network load & overhead
- Duplicate requests due to inadequate client-side caching

Flashlight Impact 🔦

- FPS → Drops while waiting for blocking resources e.g. fonts
- CPU usage → High while decoding responses
- Sluggish UI and delays in loading content

bitglow

Batch API requests to minimise round trips

💡 Takeaway no. 4

bitglow

Too many re-renders ♻️

Pitfall: Unnecessary re-renders, prop drilling & inline functions 🚩

- Too many re-renders block the main JS thread
- Expensive data manipulations in render loop

Flashlight Impact 🔦

- FPS → Drops during interactions e.g. scrolling
- CPU usage → Inefficient diffing
- Sluggish UI and poor UX

bitglow

Keep the render function lean

💡 Takeaway no. 5

bitglow

🐾 Designing & implementing a multi-phase improvement plan

bitglow

🧭 Roadmap

- Analyse codebase looking out for common pitfalls
- Identify and prioritise optimisations at feature & component level
- Integrate performance enhancements into new feature development
- Stay agile to keep up with evolving user expectations and fast-moving trends in e-commerce
- Incorporate performance metrics into release pipeline to ensure ongoing visibility, regression detection & accountability

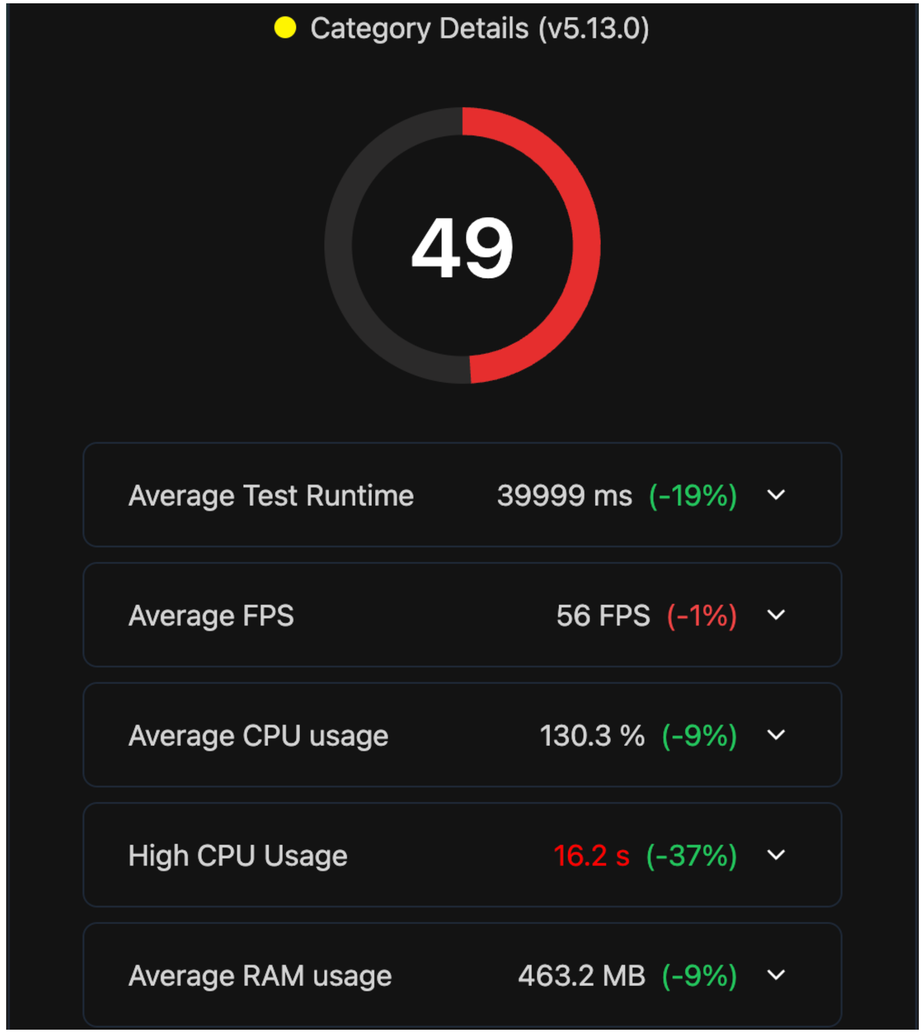
bitglow

Incrementing on Performance

bitglow

Results v5.13.0

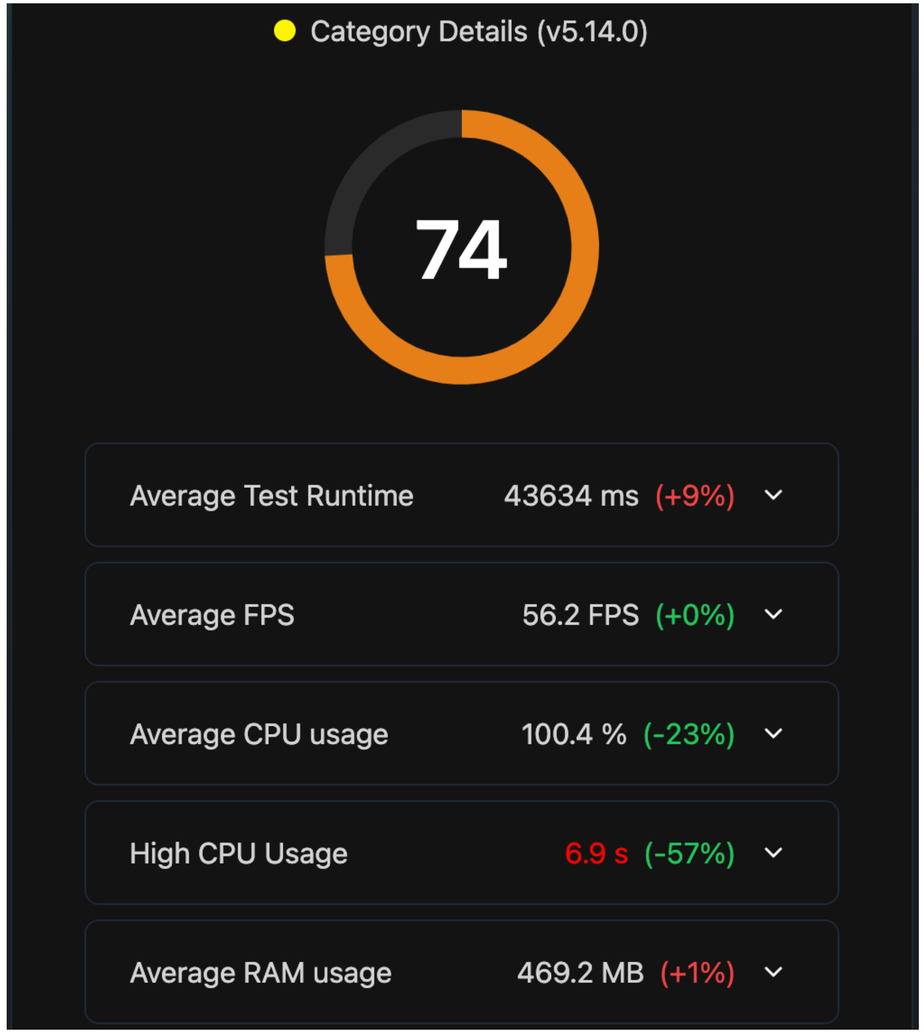
- Batched product network requests
- Lean product list model
- Tuned network request to reduce redundant data transfer



bitglow

Results v5.14.0

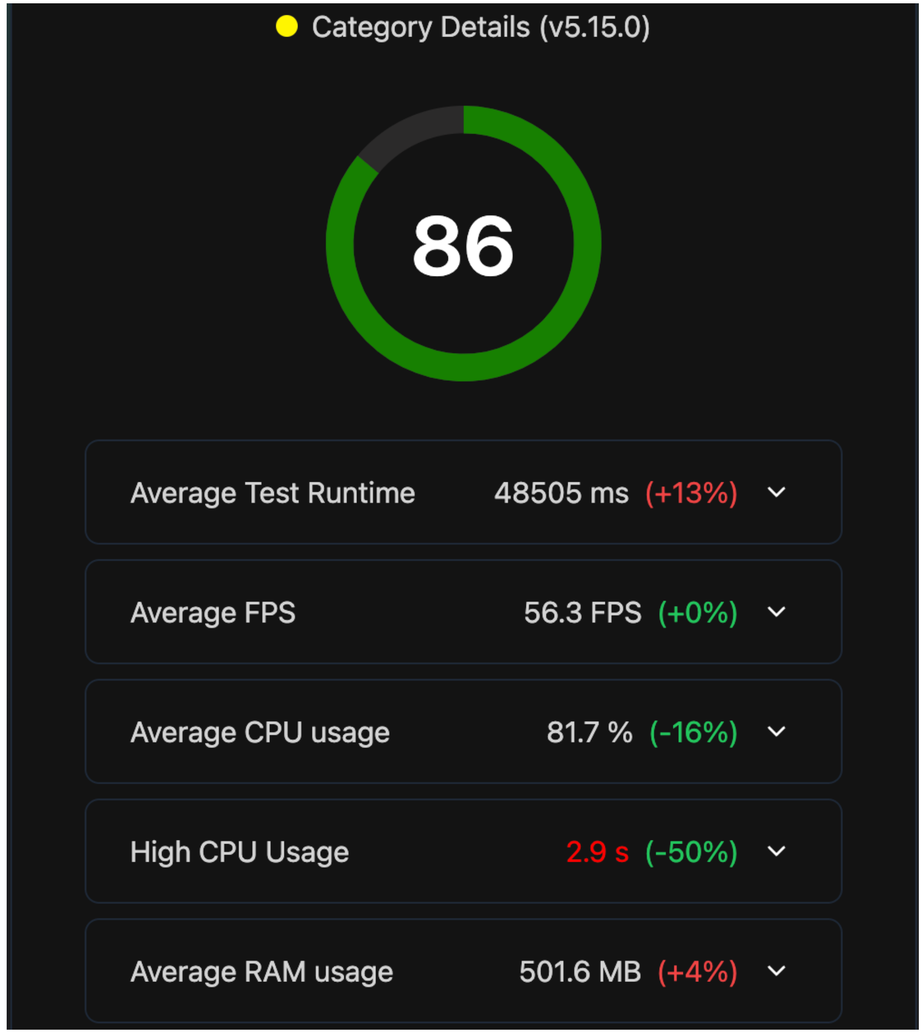
- Reduced re-renders by tweaking redux data selection



bitglow

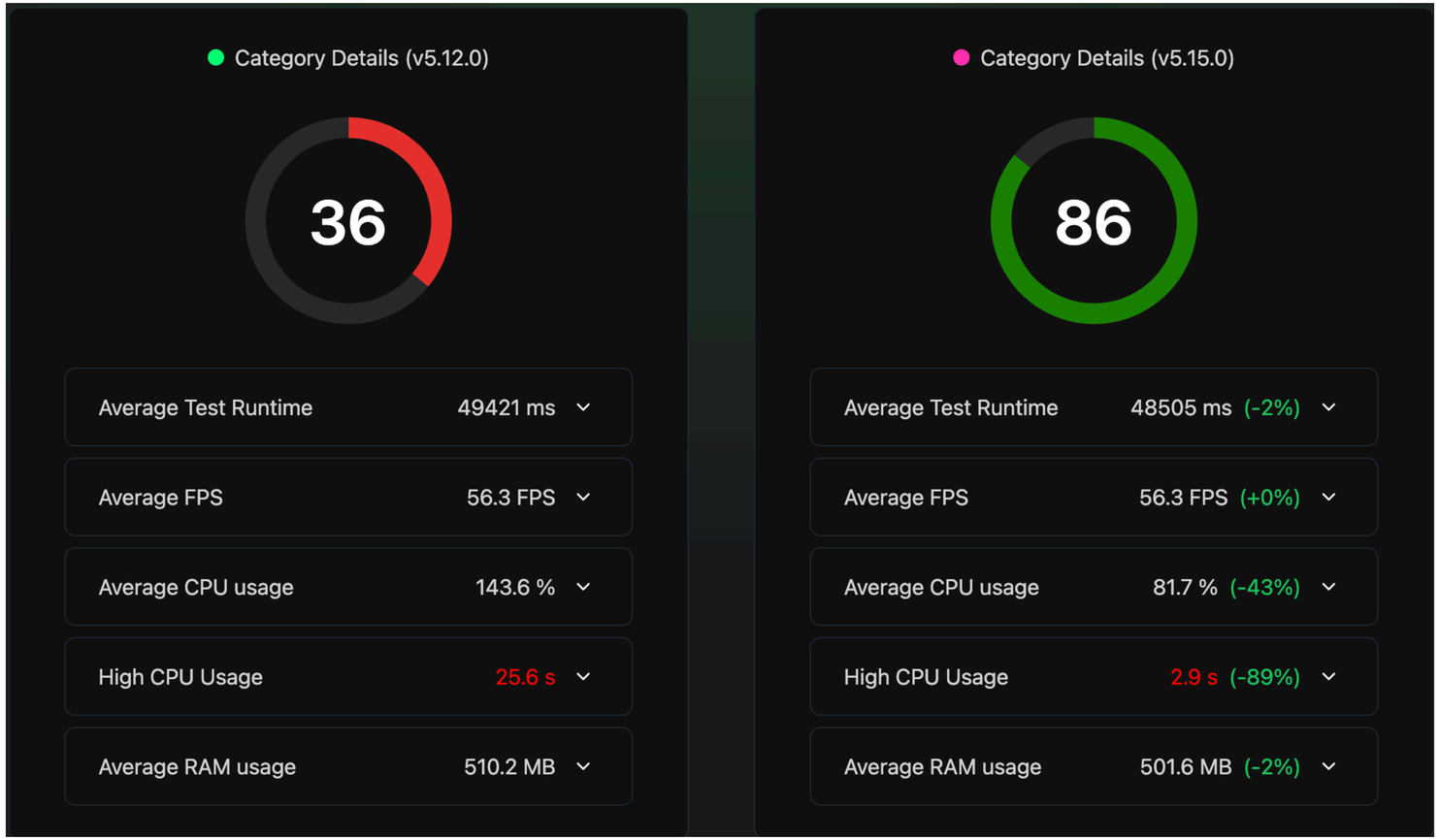
Results v5.15.0

- Reduced re-renders by removing the Dynamic App Header



bitglow

♻️ Reviewing our results

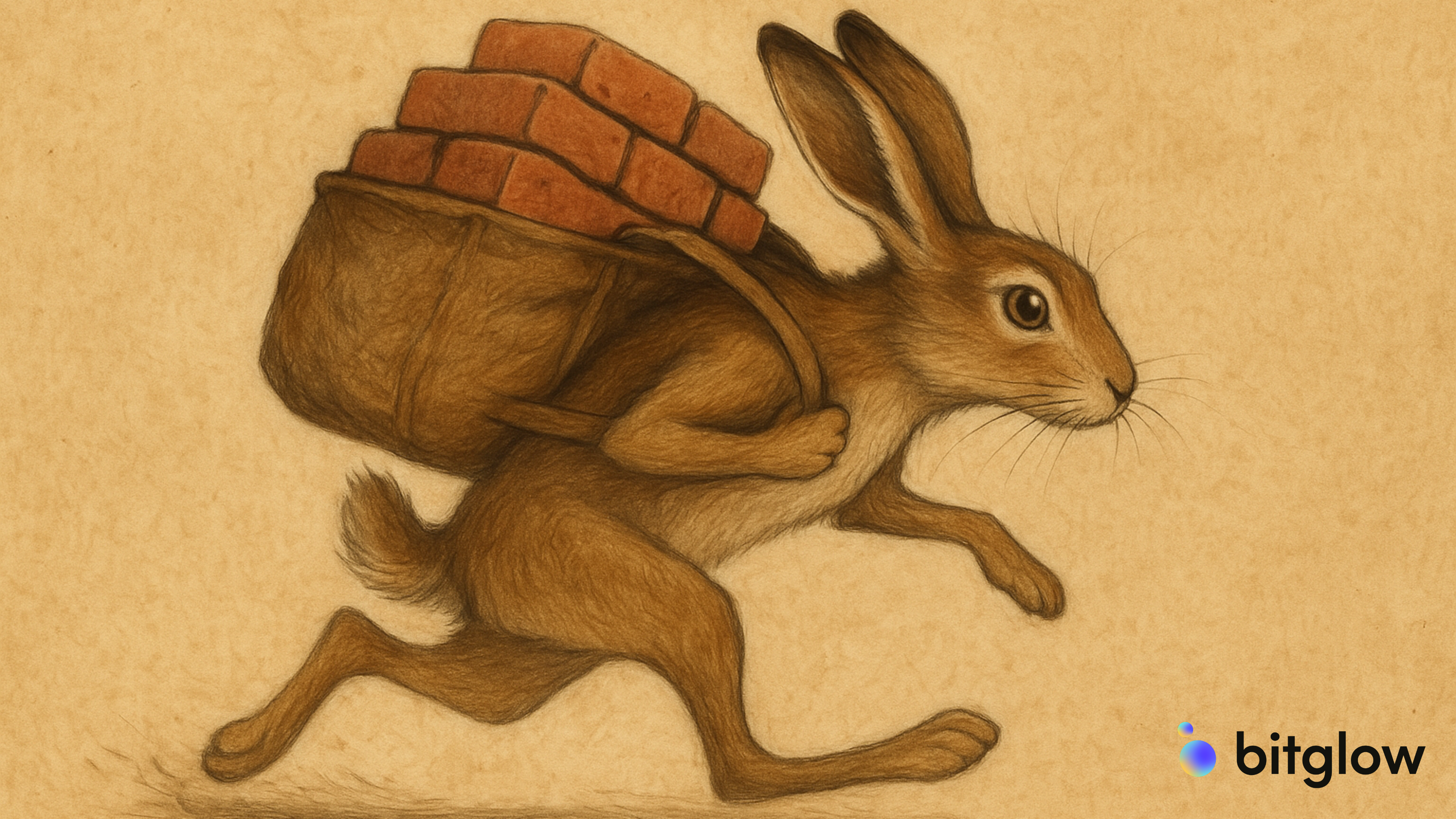


bitglow

Results over the past 12 months

	App Launch (Shop Tab)	Home Tab	Wishlist Tab	Basket Tab	Account Tab	Category Details (PLP)	Inspiration Overview	Inspiration Page	Shop the Look Page	Product Details Page (PDP)
v5.12.0	47	13	49	45	55	36	67	50	57	48
v5.13.0	76	41	73	71	78	49	87	70	77	64
v5.14.0	71	52	68	66	75	74	85	81	78	68
v5.14.1	74	63	79	72	79	77	86	81	79	67
v5.15.0	73	55	75	81	81	86	85	82	77	80
v5.16.0	75	55	79	73	77	85	83	77	76	80
v5.16.1	77	61	79	76	79	85	84	78	75	81
v5.17.0	75	61	80	79	78	86	86	81	84	83
v6.0.0	85	89	89	89	90	86	N/A	N/A	N/A	90
v6.0.1	84	90	91	91	89	90	N/A	N/A	N/A	93
v6.0.2	87	90	89	89	91	89	N/A	N/A	N/A	92
v6.0.3	87	91	91	90	92	89	N/A	N/A	N/A	93

bitglow



bitglow

Summary

Agenda

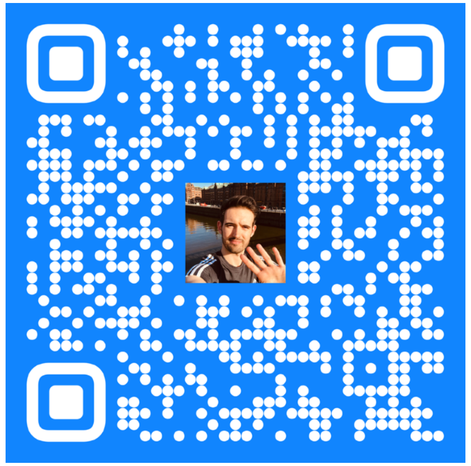
- App Performance Isn’t Just Technical - It’s Commercial
- Resize images on the server
- Only render what's necessary or soon to be visible
- Batch API requests to minimise round trips
- Keep the render function lean

That's all folks!

- 🚀 Performance audit
- 📱 Support on React Native app
- 📩 Drop us a line at: hi@bitglow.de

Jonathan Bones

@bonesyblue



bitglow