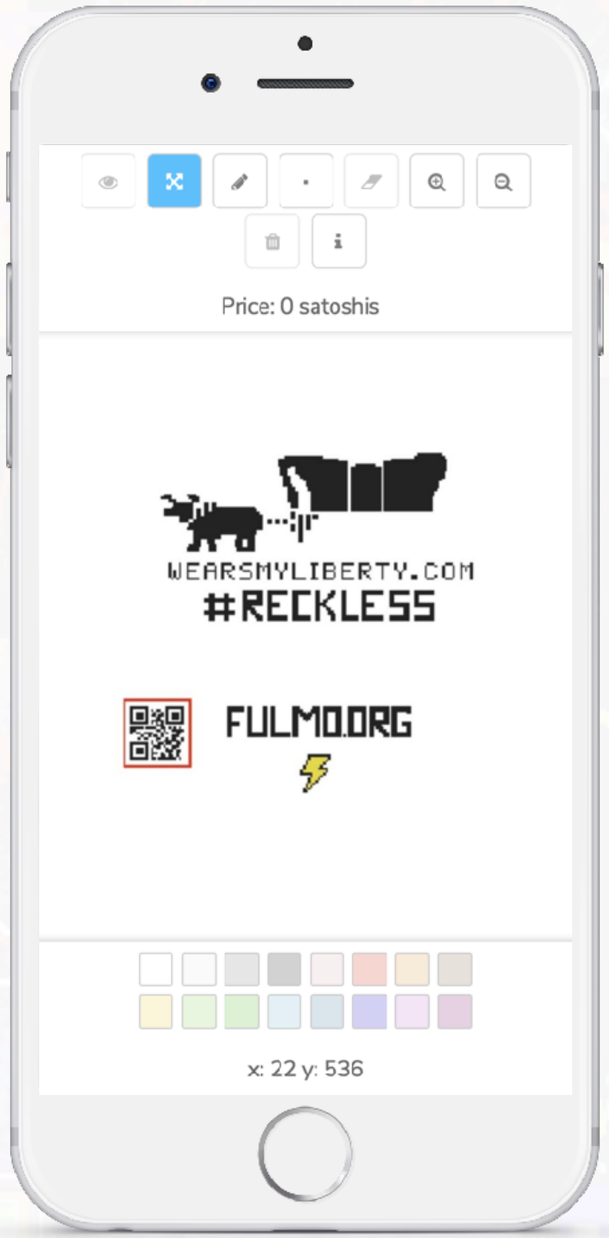
Building satoshis.place

(Part 2)



Aim for

UX

Fast
Painting updates instantly after payment.

Cross-platform
Desktop and mobile.

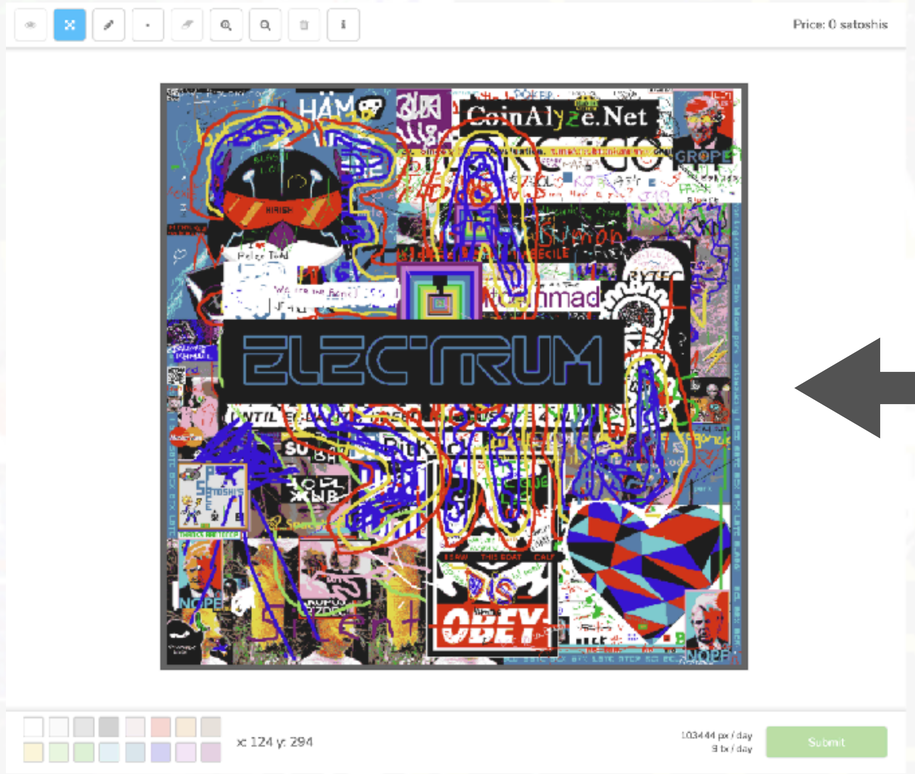
Unfairly Cheap™
Accessible to all.

Simple and Delightful
Make it intuitive and satisfying to use.

Hackable
API for developers.

Fast

1. Display canvas as an image (“sprite” in PixiJS)



Red	Green	…
0000	0001	0010

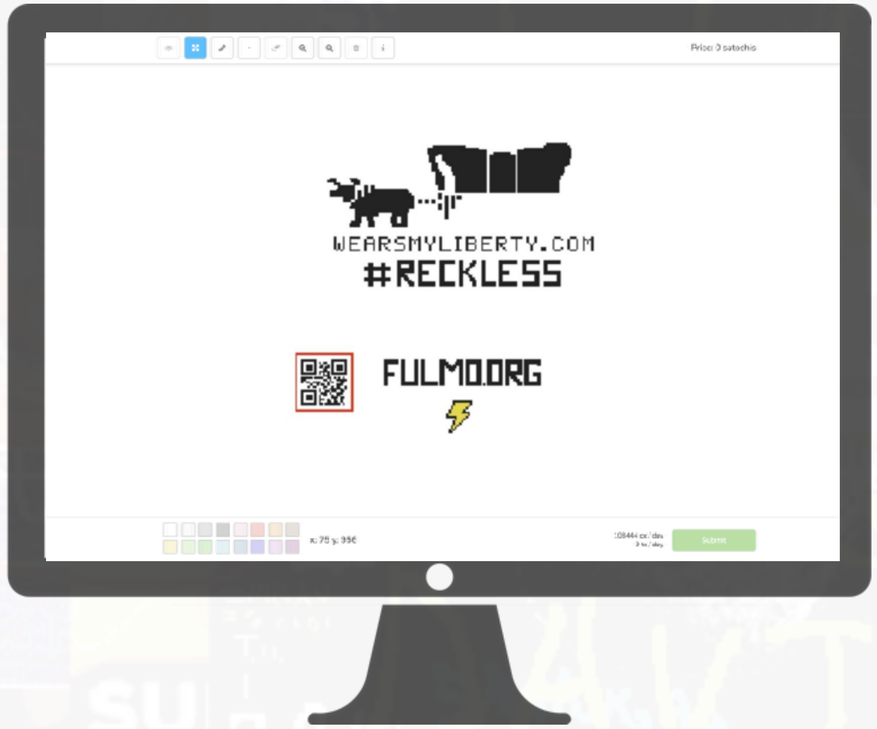
Display as PNG Image

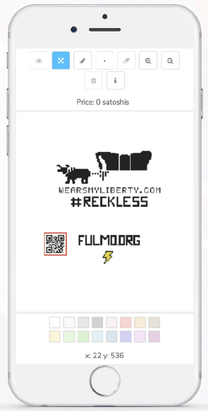
Save as base64 representation

AAADVCAMAAAAfHvCaAAAAGFBMVEVYn%2BH
%2F%2F%2F%2….

Cross-platform

1. Responsive design





2. Touch and mouse support

```
// Setup listeners
window.addEventListener('touchstart', self.setMouseDown)
window.addEventListener('touchend', self.setMouseUp)
window.addEventListener('mousedown', self.setMouseDown)
window.addEventListener('mouseup', self.setMouseUp)
```

3. Multi-touch “pinch” / “zoom” gestures

```
// Activate viewport plugins
viewport
  .decelerate({ friction: 0.8 })
  .pinch()
```

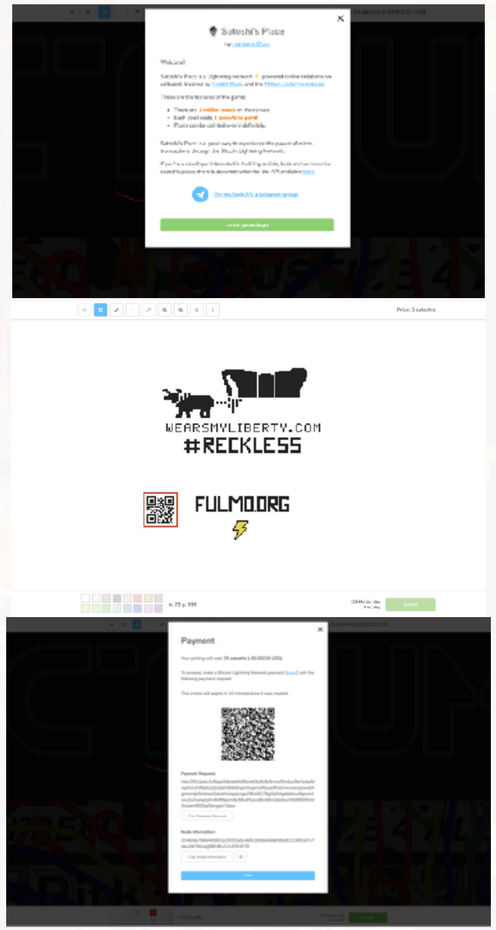
Unfairly Cheap™

1. Fixed pricing of 1 satoshi per pixel.

2. No limit on amount of times pixels can be drawn.

3. No pixel “ownership” MLM scheme.

Simple and Delightful



1. Welcome screen.

2. Avoid cluttering the UI.

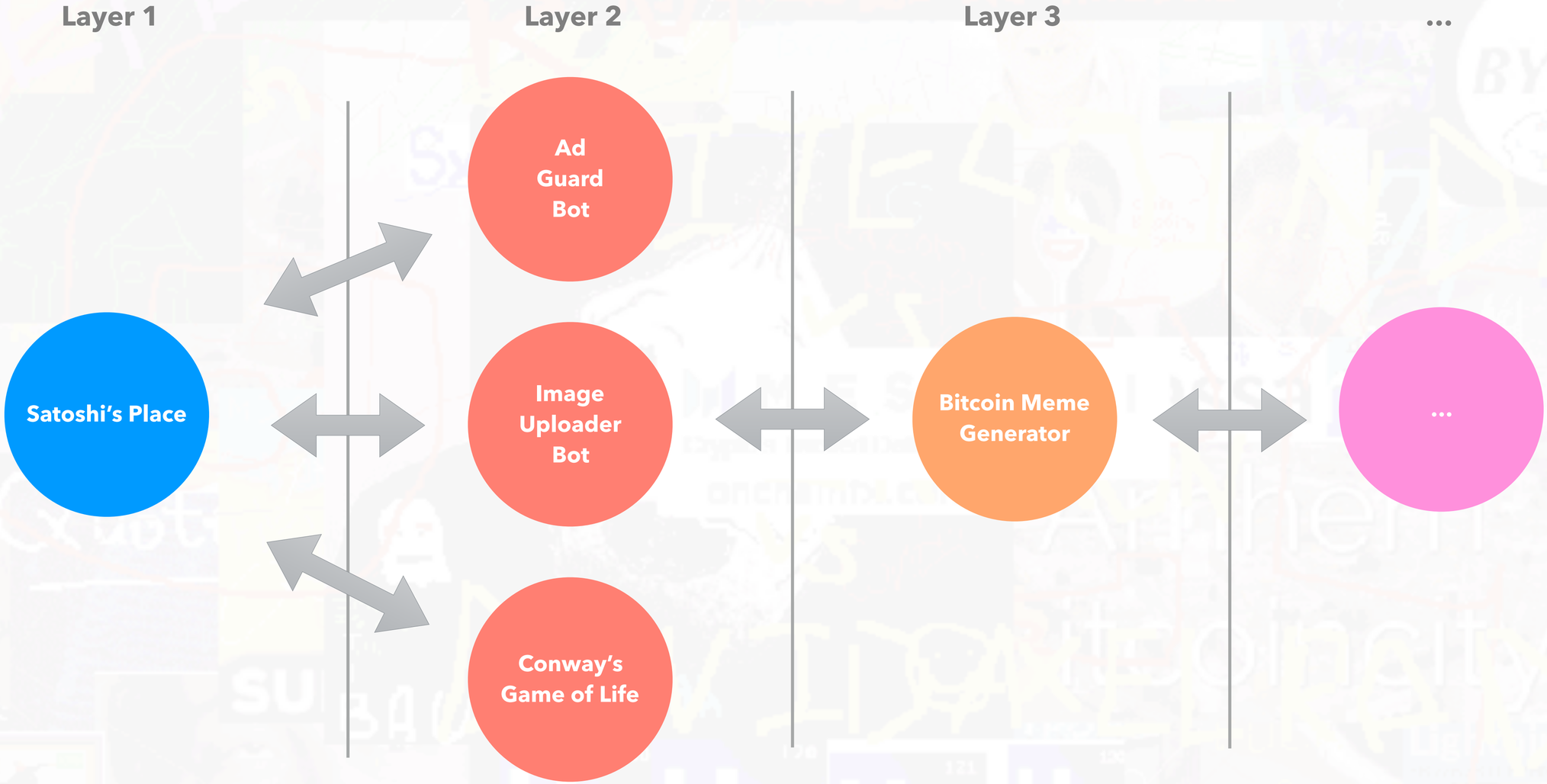
3. Minimum 3 clicks for invoice. Look for shortcuts!

4. Reactive.

5. Reduce barriers of entry.

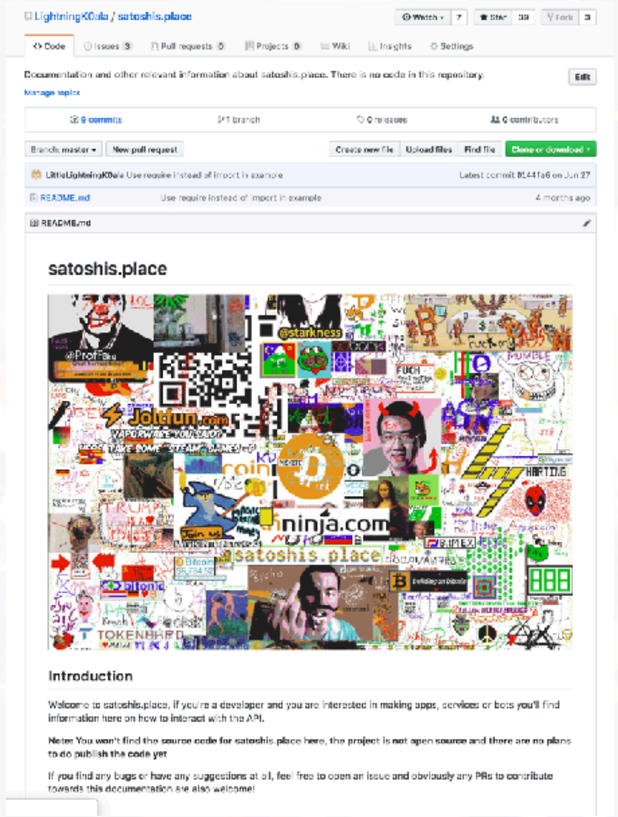
6. 16 colors.

Hackable


Layer 1
Layer 2
Layer 3
...
Satoshi's Place
Ad Guard Bot
Image Uploader Bot
Conway's Game of Life
Bitcoin Meme Generator
...

Hackable

1. Publish API documentation on github with examples.



2. Keep new order payload simple.

NEW_ORDER

When you want to draw something, send a request with this event and an array of objects like:

```
[
  {
    coordinates: [0, 0],
    color: '#ffffff'
  },
  ...
]
```

Architecture Outline

Web Server

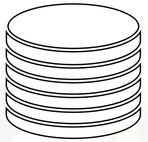
Serve client application

Backend Server

Handle business logic
and expose socket API

Lightning Node

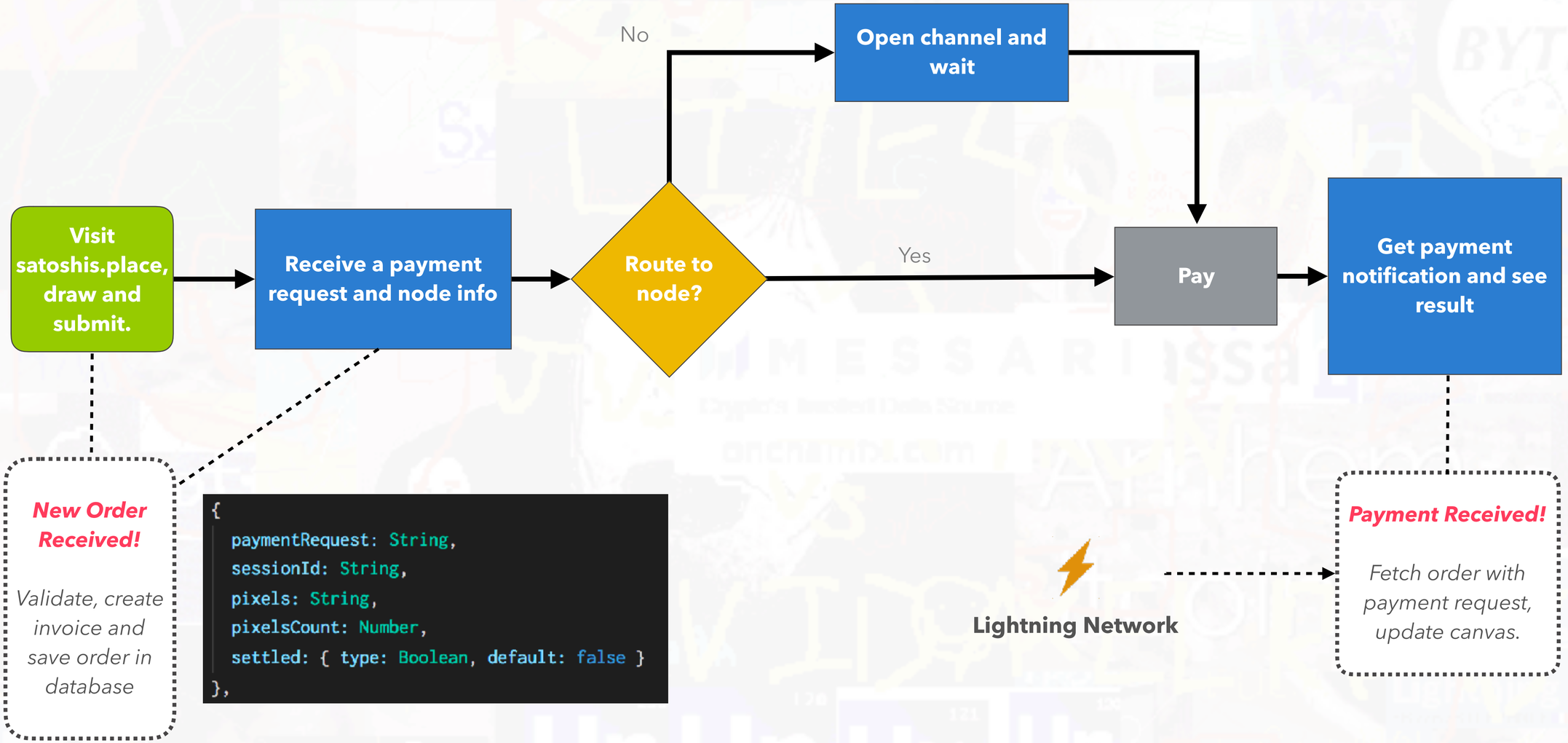
Access to the lightning
network



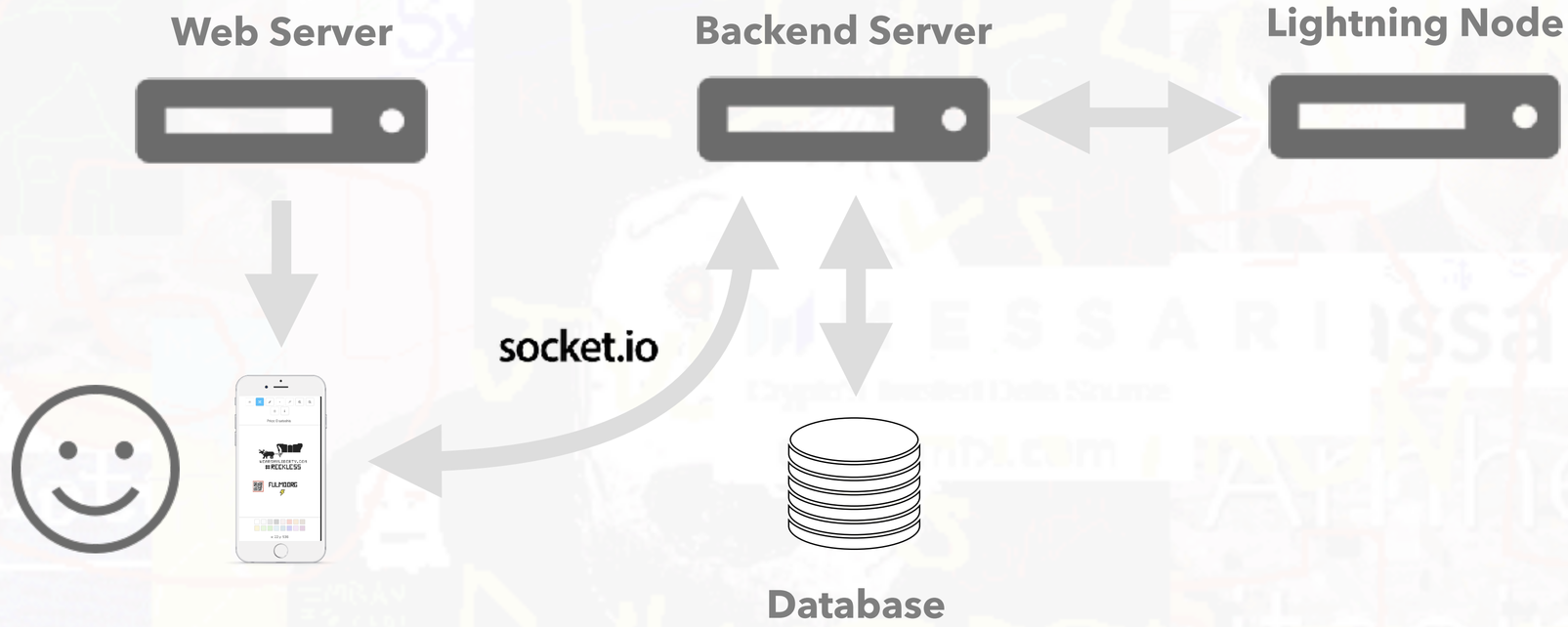
Database

Store settings, orders and base64
png representation of canvas

UX Flow Diagram



Architecture Outline



Tech Stack Overview

Web Server

NEXT.JS

PixiJS v4

styled components

]v[

Backend Server

node JS

express

Database

mongoDB

Lightning Node

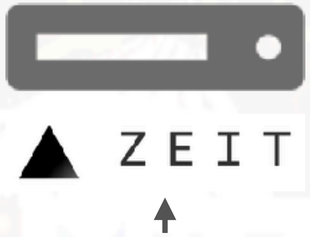
Bitcoin Core

Blockstream

Hosting

Web Server

Backend Server



Dynamically scaled instances to cope with increasing traffic

Lightning Node

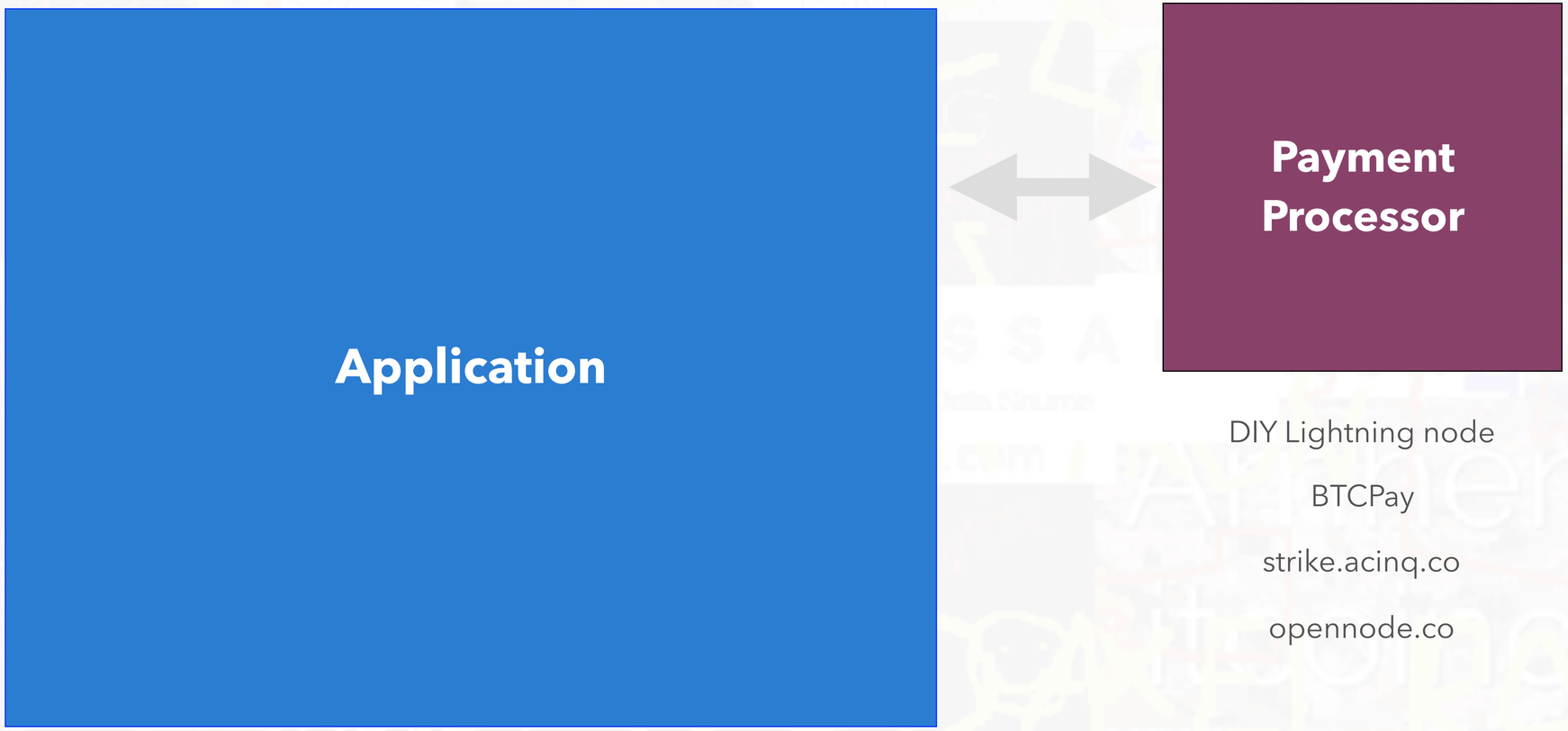
Scaleway (testnet)

ODROID XU4 (mainnet)

Database

mLab

Lightning App Mental Model



Optimizations

Encoding pixels to PNG
Represent the entire canvas as an image. Optimizes performance, storage and bandwidth usage.

Prune old orders
Set invoice expiry (i.e. 1 hour) and prune non paid orders at the end of the day.

Request limit API
Track number of requests coming from an IP and enforce limit (i.e. 10 per minute).
Enforce max size order.

What worked (1/2)

Word of mouth

Sharing the project via direct message to influential people offering them to beta test.

SBC Lightning Node

Your own private payment processor at home.

Eclair Wallet

Convenient spend-only mobile mainnet + testnet lightning wallet.

Testnet

Use testnet / regtest to prototype.
Launching with testnet might also be useful for users to experiment with your app.

What worked (2/2)

Daily Database Backups

Keeping a historical record of all the orders made whilst reducing storage requirement on the database hosting service.

Websockets

Good way to create a reactive experience.

Docker

Create a local replicable working environment that can be easily deployed to production.

Help! I need somebody!

Don’t be afraid to reach out to other developers member of the community for help

API Documentation

A number of third party tools and bots were made. A lot of image uploads by the community.

What didn't work

Telegram Group

Hands-off moderation of Telegram group led to some toxicity in the chat, alienating constructive discussions about lightning and driving users away.

Saving image to database

Would've been better to save canvas as an image in S3 to save bandwidth.

Missing message broker and payment watcher

A message broker is needed if the API is scaled to multiple instances so that client <-> backend messages can be routed properly. A payment watcher is needed to process the payment only once.

Takeaways

1. Iterate on your initial ideas.
2. Keep it usable.
3. Keep it simple.
4. Deploy early.
5. Collaborate.
6. Be CRAEFUL.
6. Have fun!

Questions?