Legend: In progress DONE

1. Redesigned frontend (#1899)
    a. Establish frontend design process
    b. Re-design the overall experience
        i. Index page: show something more than just the name of the category. Give an example. Things like … Or a sentence of descriptions.
    c. Better modal (#2496) 🚀1/10
    d. Add UI elements for the remaining badge options (#2867)
        i. Logo
        ii. Cache seconds
    e. Adopt Gatsby (#1943)
    f. Fix frontend code coverage reporting (#2616)
    g. Document badge best practices (#2868)
        i. Show examples of prominent users of Shields badges
    h. Page for each service family
    i. Drop-downs for the multiple-choice components of badges (#2882)
    j. Suggested-badge improvements
        i. Create badge URL from project URL (#2420)
        ii. Handle suggestions for URLs from npmjs, gitlab, travis etc. (#2679)
    k. Tag new badges with “new” indicator
    l. New visual design
2. New badge features
    a. Endpoint badge 🚀1/21
    b. Support for Gitlab (#541)
    c. Split “quality” into “code coverage” and “analysis” 🚀1/10
3. Rework the way we handle logos (#2510)
    a. Automatic light/dark logo colors for simple-icons (#2431 #2833)
    b. Don’t recolorize multi-colored Shields icons (#2889)
    c. Remove unnecessary logos (#2857)
    d. Update logo contributing guidelines (#2888)
4. Modernize the badge API
    a. Query-param based static badge interface (#3024)
    b. New URL-based static badge interface (#2673)
    c. Redirect deprecated badge URLs to canonical URLs (#2340)
    d. Harmonize color param names with Endpoint badge (#3012)
    e. Harmonize cache param name with Endpoint badge (#3069)

    f. svg by default (#2674)
    g. Deprecate raster badges or move to a separately deployed service (#3112)
    h. New JSON format (#2686)

5. Stability & security
    a. Rewrite and test Github auth logic, separating standard and search quota (#1205) 🚀 1/10
    b. Limit the size of responses we will accept 🚀1/11
    c. Lock down services which accept URLs to require https where possible (#2637)
    d. Require production traffic to pass through the CDN (#3027)
    e. Require https on the origin servers
6. Deployment (#1848)
    a. PaaS, CI, and production-friendly config 🚀1/6
    b. Cloud-backed persistence for GitHub tokens (#3332)
    c. Bring a fourth server online and/or migrate to a PaaS and scale up
    d. Get package-lock working (#1988)
    e. Automate the deployment validation process (i.e. smoke tests)
    f. Automate the deployment process
    g. Evaluate other CDNs (#3026)
    h. Upgrade to Node 10/12 (#3403)
7. Ops foundation
    a. Document a security program (#2577)
    b. Make sure at least two active maintainers have access to every aspect of the project (see table)
8. Adopt a code of conduct (#2671)
9. Analytics (#2068)
    a. Response time by service
    b. Usage by service (#966)
    c. More detailed response time information using NodePing
10. Finish refactoring services (#2863)
11. Improve the general developer experience
    a. CircleCI test summary for daily tests 🚀1/6
    b. CircleCI stability and performance improvements 🚀1/21
    c. Reorganize all the library and helper code (#2832)
    d. Implement a merge queue for pull requests (#2807)
    e. npm start with autoreload (#2426)
    ~~f. Speed up CI builds using workspace caching (#1937)~~
    g. Report number of passing + failing service tests (#2642)
    h. Update config for Mocha 6 (#2619)

i. Print a danger warning when modified services are not in the PR title (#2545)

12. Make BaseService easier to use and understand
    a. Validate return values and class properties
        i. Validate services on load (#2769)
        ii. Validate return value from handle() (#2740)
        iii. Validate route property (#2860)
        iv. Validate example keywords (#2956)
        v. Validate calls to _request, _requestJson, etc.
        vi. others?
    b. Make the code more readable
    c. Autocomplete-friendly documentation (#3645 and others)
    d. Validate query parameters within BaseService (#2676)
    e. Support error schema (#2632)
    f. Reduce complexity
        i. Finish converting static previews (#1961, #2479)
        ii. Consolidate service definition schema, removing support for previewUrl (#2895)
        iii. Remove dependency on legacy request handler (#3368)
            1. Implement new timeout
            2. Remove in-memory caching or implement new in-memory cache
        iv. Deprecate route `format`, replacing with `pattern` (#3329)
        v. Refactor logo resolution so it happens in the badge generator (#3370)
        vi. Bring serviceData and makeBadge schemas into full alignment (#3370)
    g. 100% unit test coverage
    h. Move route functions into their own module (#2860)
    i. Detect overlapping routes (#2155)
13. Make badge helper functions easier to use and understand
    a. Autocomplete-friendly documentation
    b. 100% unit test coverage
14. Make the badge generator easier to understand
    a. Move badge templates from doT → Template literals (#2428)
15. Make the frontend easier to understand
    a. Rewrite class components as function components using React Hooks
        i. DynamicBadgeMaker, StaticBadgeMaker (#3096)
        ii. BadgeExample, CategoryHeading, etc. (#3177)
        iii. CopiedContentIndicator (#3719)
        iv. Customizer (#3720)

        v. PathBuilder (#3721)
        vi. QueryStringBuilder (#3740)
        vii. SuggestionAndSearch (#3741)
        viii. RequestMarkupButton (#3739)
        ix. Main (#3738)
    b. Adopt TypeScript (#3657)
16. Amp up general project documentation
    a. Document server secrets 🚀1/2
    b. Document production hosting 🚀1/7
    c. High-level code walkthrough (#2684)
    d. Front-end tooling + library walkthrough
    e. Supported deployment environments
    f. Document how to make redirects (#3402)
    g. Tips, contributing guidelines, design guidelines, etc. (draft)
    h. Establish design process for badges
    i. Privacy policy (#4225)
17. Self-hosting improvements
    a. Changelog
    b. Release plan
18. Show our sponsors on the website
19. Consider using bounties to accelerate the roadmap

Later

1. Distributed hack day for new badges
    a. Partner with Gitpod (#2772)
2. Rework server using a framework that is well documented and tested
3. Create new service testing workflow which invokes the code directly instead of over HTTP
4. Caching API responses (#1545)
5. Caching layer for discardable credentials (#2547)
6. Better debug experience
    a. Better logging for BaseService
        i. Only trace failures
    b. A trace web endpoint that lets you see all the trace output for a badge on a production server
    c. A trace CLI: run one badge and spit out the trace results