Demo: Building Ephemeral Notes

▼ Prompt 1: The Core UI (Preventing Serialization Errors)

@apps/frontend/src/routes/index.tsx

I am building a "Secret Note" feature.
Task: Modify the Landing Page component.

1. State: Add a Qwik Signal to toggle between "Shorten URL" and "Create Secret Note".
2. UI: Add a visual toggle above the main input.
3. Logic (Secret Note Mode): - Show a `<textarea>` for the user to type.
    - On input, use `lz-string` to compress the text.
    - Update the main URL input field with: `${window.location.origin}/note#${compressed_string}`.
    - Make the main input read-only in this mode.

Critical Constraints: - Wrap the compression helper function in a `$(...)` QRL to prevent Qwik serialization errors.

- Ensure `lz-string` is imported correctly.

▼ Prompt 2: The Note Reader (Hydration & Security)

Create a new route: @apps/frontend/src/routes/note/index.tsx

Task: Build the "Read Note" page.

1. Hydration Safety: Use `useVisibleTask$` to access `window.location.hash`. DO NOT read the hash in the main component body (Server crash risk).
2. Logic: Strip the leading `#`, decompress using `lz-string`, and store in a signal.
3. UI: Display the message in a centered card. Add a "Reply" button (link to home).

4. Security: Render the decoded text as plain text strings only. Do NOT use `dangerouslySetInnerHTML`.
5. Error Handling: If decompression returns null, show a friendly "Note is broken" message.

▼ Prompt 3: The Frontend Validation Jailbreak

@apps/frontend/src/routes/index.tsx

I need to relax the frontend validation to allow these internal "Note" URLs.

Task:

1. Unlock the Button: Update the "Shorten" button's `disabled` attribute. It must be ENABLED if `isSecretNoteMode` is true, regardless of the regex check.
2. Unlock the Handler: In the `onSubmit$` (or similar) handler, find the guard clause `if (!isValid) return;`. Update it to allow execution if `isSecretNoteMode` is true.

Goal: Ensure the form actually submits the localhost URL to the backend action.

▼ Prompt 4: The Backend Logic (Data Transfer Object Validation & Idempotency)

@apps/backend

I need to adjust the backend to accept internal URLs and handle duplicates gracefully.

Task 1: Relax DTO Validation

- Find the DTO for creating a link (likely `create-link.dto.ts`).
- Update the URL validator to allow `localhost` and missing TLDs (e.g., `@IsUrl({ require_tld: false })`).

Task 2: Idempotency

- Find the `create` method in the Link Service.
- If a link already exists (duplicate), do NOT throw an error.
- Instead, catch the specific conflict/duplicate error and return the existing link record.

Goal: The API should never fail on valid input, even if it's a duplicate.

▼ Prompt 5: Fixing the “URL Already Shortened” Error

I’m getting an error saying that the URL is already shortened. I suspect it’s because the backend is treating any URL containing our own domain as “already shortened”. Can you please fix this by skipping validation for paths starting with “/note”?

▼ Prompt 6: The Safety Polish (Redirects & Tests)

Two final polish tasks:

1. Redirect Safety: In the redirection component (likely `apps/frontend/src/routes/[id]/index.tsx`), ensure we don't double-protocol the URL.
    - Check: `if (url.startsWith('http')) window.location.href = url;`
    - Else: `window.location.href = 'https://' + url;`
2. Unit Test: Create `@libs/core/src/note-compression.spec.ts`.
    - Test that `lz-string` compression/decompression is deterministic with special characters.

▼ Prompt 7: (Optional) Make Secret Message More User Friendly by Selecting Secret Message When Reply is Hit

Finally, when I open a secret message I see the button to reply. But when I hit reply, I'm taken back to the homepage with the URL shortener automatically selected. Can you please change the logic to have the secret message state selected automatically when a user hits reply?