Feature-Ideen für Best of Games

1. Kuratoren-Listen (Curator Lists)

Status: Geplant
Geschätzter Aufwand: ~4-5 Stunden

Kuratierte Sammlungen wie "Top 10 Cozy Games", "Beste Roguelikes unter 15€", "Hidden Gems 2024".

Warum dieses Feature?

Listen sind **Traffic-Magnete** weil sie Long-Tail Keywords abdecken, die einzelne Spieleseiten nicht erreichen:

- "beste roguelikes unter 15 euro"
- "cozy games 2024"
- "hidden gems indie spiele"

Architektur

Dynamisch vs. Statisch

Ansatz	Speichert	Aktualisierung	Empfohlen
Statisch	game_ids	Nur bei Cron-Lauf	Nein
Dynamisch	criteria (Regeln)	Bei jedem Request	Ja ✅

Dynamisch bedeutet:

- Neues Roguelike wird hinzugefügt → erscheint sofort in "Top Roguelikes"
- "Beste Spiele dieser Woche" zeigt immer die letzten 7 Tage
- Kein Cron-Job nötig für Listen-Updates

Datenbank-Schema

```
// src/lib/schema.ts

export const curatorLists = pgTable('curator_lists', {
  id: uuid('id').defaultRandom().primaryKey(),
  slug: text('slug').notNull().unique(),
  title: jsonb('title').$type<LocalizedField>().notNull(),
  description: jsonb('description').$type<LocalizedField>(),
  criteria: jsonb('criteria').$type<ListCriteria>().notNull(),
  coverImage: text('cover_image'),  // Optional: Hero-Bild für Liste
  sortOrder: integer('sort_order').default(0),
  isActive: boolean('is_active').default(true),
  isFeatured: boolean('is_featured').default(false),  // Für Homepage
  createdAt: timestamp('created_at', { withTimezone: true }).defaultNow(),
  updatedAt: timestamp('updated_at', { withTimezone: true }).defaultNow(),
});
```

Kriterien-Typ

```
// src/lib/list-criteria.ts

export type ListCriteria = {
  // Tag-Filter (OR-Verknüpfung)
  tags?: string[];

  // Score-Filter
  minScore?: number;
  maxScore?: number;

  // Preis-Filter (in Cents)
  maxPrice?: number;
  minDiscount?: number;
  onSale?: boolean;

  // Zeit-Filter (RELATIV - immer aktuell!)
```

```
  publishedWithinDays?: number;   // Review veröffentlicht innerhalb X
Tage
  releasedWithinDays?: number;    // Spiel released innerhalb X Tage

  // Zeit-Filter (ABSOLUT - fixiert)
  year?: number;

  // Sortierung
  orderBy?: 'score' | 'publishedAt' | 'releaseDate' | 'title';
  orderDirection?: 'asc' | 'desc';

  // Limit
  limit?: number;
};
```

Beispiel-Listen

Slug	Titel (DE)	Kriterien
top-roguelike-games	Top 10 Roguelikes	{ tags: ["Roguelike"], orderBy: "score", limit: 10 }
best-this-week	Beste Spiele der Woche	{ publishedWithinDays: 7, orderBy: "score", limit: 10 }
new-this-month	Neu diesen Monat	{ publishedWithinDays: 30, orderBy: "publishedAt" }
best-cozy-games	Top 10 Cozy Games	{ tags: ["Cozy"], orderBy: "score", limit: 10 }
games-under-10-euros	Beste Spiele unter 10€	{ maxPrice: 1000, minScore: 7.0 }
games-under-15-euros	Beste Spiele unter 15€	{ maxPrice: 1500, minScore: 7.0 }
best-of-2024	Beste Indie-Spiele 2024	{ year: 2024, orderBy: "score", limit: 20 }
hidden-gems	Geheimtipps	{ minScore: 7.5, maxScore: 8.5, limit: 15 }
on-sale-now	Aktuell im Sale	{ onSale: true, minScore: 7.0 }

Zeitbasierte Listen - So funktioniert's

```
Montag, 9. Dezember - User besucht "Beste Spiele der Woche":
┌──────────────────────────────────────┐
│ Beste Spiele der Woche               │
│ (Reviews vom 2.-9. Dezember)         │
```

```
┌─────────────────────────────────────────────┐
│ 1. Game A (Score: 9.2) – 5. Dez             │
│ 2. Game B (Score: 8.8) – 7. Dez             │
│ 3. Game C (Score: 8.5) – 3. Dez             │
└─────────────────────────────────────────────┘

Eine Woche später, Montag 16. Dezember:

┌─────────────────────────────────────────────┐
│ Beste Spiele der Woche                      │
│ (Reviews vom 9.–16. Dezember)               │  ← Automatisch verschoben!
├─────────────────────────────────────────────┤
│ 1. Game D (Score: 9.5) – 12. Dez            │  ← Neue Spiele!
│ 2. Game E (Score: 9.0) – 10. Dez            │
│ 3. Game A (Score: 9.2) – 5. Dez             │  ← Fällt nächste Woche raus
└─────────────────────────────────────────────┘
```

Query-Funktionen

```
// src/lib/queries.ts

// Alle aktiven Listen (für /lists Übersicht)
export async function getCuratorLists(): Promise<CuratorList[]>

// Featured Listen (für Homepage)
export async function getFeaturedLists(): Promise<CuratorList[]>

// Einzelne Liste laden
export async function getCuratorListBySlug(slug: string):
Promise<CuratorList | null>

// Spiele basierend auf Kriterien – DYNAMISCH!
export async function getGamesForList(criteria: ListCriteria):
Promise<ReviewListItem[]>
```

`getGamesForList` baut dynamisch eine SQL-Query:

```
export async function getGamesForList(criteria: ListCriteria):
Promise<ReviewListItem[]> {
  const conditions = [eq(reviews.isPublished, true)];

  // Relativer Zeitfilter – wird bei JEDEM Request neu berechnet!
  if (criteria.publishedWithinDays) {
    conditions.push(
      sql`${reviews.publishedAt} >= NOW() - INTERVAL
'${criteria.publishedWithinDays} days'`
    );
  }
```

```
  if (criteria.minScore) {
    conditions.push(sql`CAST(${reviews.score} AS DECIMAL) >=
${criteria.minScore}`);
  }

  if (criteria.tags?.length) {
    // Join mit tags Tabelle
  }

  if (criteria.maxPrice) {
    // Join mit price_snapshots (latest)
  }

  // ... weitere Bedingungen

  return db
    .select({ /* ... */ })
    .from(reviews)
    .innerJoin(games, eq(reviews.gameId, games.id))
    .where(and(...conditions))
    .orderBy(/* basierend auf criteria.orderBy */)
    .limit(criteria.limit || 20);
}
```

Frontend-Seiten

1. Listen-Übersicht: `/[lang]/lists/page.tsx`

- Zeigt alle aktiven Listen als Karten
- Jede Karte: Titel, Beschreibung, Anzahl Spiele, Cover-Bild
- Link zu `/lists/[slug]`

2. Listen-Detail: `/[lang]/lists/[slug]/page.tsx`

- Lädt Liste via `getCuratorListBySlug`
- Lädt Spiele via `getGamesForList(list.criteria)`
- Rendert Spiele mit `ReviewCard` Komponente
- SEO: Meta-Tags, Structured Data (ItemList Schema)

3. Homepage-Integration: `/[lang]/page.tsx`

- Neue Sektion "Featured Lists" nach Recent Reviews
- Zeigt 3-4 Listen mit `isFeatured=true`
- Link zu `/lists` für alle Listen

Neue Komponente: `ListCard.tsx`

```
// src/components/ListCard.tsx

type ListCardProps = {
  slug: string;
  title: string;
  description?: string;
  coverImage?: string;
  gameCount: number;
  locale: Locale;
};

// Zeigt:
// - Cover-Bild oder generiertes Grid aus Spiele-Bildern
// - Titel
// - Beschreibung (gekürzt)
// - Badge mit Anzahl Spiele
// - Link zu /lists/[slug]
```

SEO

Structured Data (ItemList Schema)

```
// src/lib/structured-data.ts

export function generateCuratorListStructuredData(
  list: CuratorList,
  games: ReviewListItem[]
) {
  return {
    '@context': 'https://schema.org',
    '@type': 'ItemList',
    name: list.title,
    description: list.description,
    url: `${SITE_URL}/lists/${list.slug}`,
    numberOfItems: games.length,
    itemListElement: games.map((game, index) => ({
      '@type': 'ListItem',
      position: index + 1,
      url: `${SITE_URL}/games/${game.slug}`,
      name: game.title,
    })),
  };
}
```

Sitemap erweitern

```
// src/app/sitemap.ts

// Listen hinzufügen
const lists = await db
  .select({ slug: curatorLists.slug, updatedAt: curatorLists.updatedAt })
  .from(curatorLists)
  .where(eq(curatorLists.isActive, true));

for (const list of lists) {
  sitemapEntries.push({
    url: `${base}/${locale}/lists/${list.slug}`,
    lastModified: list.updatedAt || new Date(),
    changeFrequency: 'weekly' as const,
    priority: 0.8,  // Hoch! Listen sind wertvoller Content
  });
}
```

Interne Verlinkung

```
Homepage
├── "Entdecke unsere Listen" Sektion
│    └── Links zu Featured Lists
└── Footer: Beliebte Listen

Game Detail Page
└── "Diese Listen enthalten {Game}" (optional, später)

/lists Übersichtsseite
└── Alle Listen mit Vorschau
```

Benötigte Dateien

Aktion	Datei	Beschreibung
Ändern	`src/lib/schema.ts`	curatorLists Tabelle hinzufügen
Neu	`src/drizzle/0005_*.sql`	Migration
Neu	`src/lib/list-criteria.ts`	ListCriteria Type
Ändern	`src/lib/queries.ts`	4 neue Query-Funktionen
Ändern	`src/lib/structured-data.ts`	ItemList für Listen
Neu	`src/app/[lang]/lists/page.tsx`	Listen-Übersicht
Neu	`src/app/[lang]/lists/[slug]/page.tsx`	Listen-Detail
Ändern	`src/app/[lang]/page.tsx`	Featured Lists Sektion

Aktion	Datei	Beschreibung
Neu	`src/components/ListCard.tsx`	Vorschau-Karte
Ändern	`src/app/sitemap.ts`	Listen-URLs hinzufügen
Ändern	`src/dictionaries/en.json`	Übersetzungen
Ändern	`src/dictionaries/de.json`	Übersetzungen

Reihenfolge der Implementation

1. Schema + Migration erstellen
2. ListCriteria Type definieren
3. Query-Funktionen implementieren
4. Listen-Detail-Seite (`/lists/[slug]`)
5. Listen-Übersicht (`/lists`)
6. ListCard Komponente
7. Homepage-Integration (Featured Lists)
8. SEO (Structured Data, Sitemap)
9. Seed-Daten: Erste Listen via SQL einfügen

Listen via SQL erstellen (Beispiele)

```
-- Top Roguelikes
INSERT INTO curator_lists (slug, title, description, criteria,
is_featured, sort_order)
VALUES (
  'top-roguelike-games',
  '{"en": "Top 10 Roguelike Games", "de": "Top 10 Roguelike-Spiele"}',
  '{"en": "The best roguelike games, ranked by our review scores.", "de":
"Die besten Roguelike-Spiele, sortiert nach unserer Bewertung."}',
  '{"tags": ["Roguelike"], "orderBy": "score", "limit": 10}',
  true,
  1
);

-- Beste Spiele der Woche
INSERT INTO curator_lists (slug, title, description, criteria,
is_featured, sort_order)
VALUES (
  'best-this-week',
  '{"en": "Best Games This Week", "de": "Beste Spiele der Woche"}',
  '{"en": "Top-rated games from the past 7 days.", "de": "Top-bewertete
Spiele der letzten 7 Tage."}',
  '{"publishedWithinDays": 7, "orderBy": "score", "limit": 10}',
  true,
  0
);
```

```
-- Beste Spiele unter 10€
INSERT INTO curator_lists (slug, title, description, criteria,
is_featured, sort_order)
VALUES (
  'games-under-10-euros',
  '{"en": "Best Games Under €10", "de": "Beste Spiele unter 10€"}',
  '{"en": "Great games that won\''t break the bank.", "de": "Tolle
Spiele, die das Budget schonen."}',
  '{"maxPrice": 1000, "minScore": 7.0, "orderBy": "score"}',
  false,
  10
);
```

Optionale Erweiterungen (später)

Idee	Aufwand	Nutzen
Manuelles Override - einzelne Spiele hinzufügen/entfernen	Mittel	Für "Editor's Picks"
List Cover Auto-Generate - Grid aus ersten 4 Spiele-Bildern	Gering	Schönere Vorschau
"Diese Listen enthalten {Game}" auf Game-Detail-Seite	Gering	Bessere Verlinkung
Admin-UI zum Erstellen von Listen	Hoch	Benutzerfreundlicher

---

2. Preisverlauf & Charts

Status: Idee
Geschätzter Aufwand: ~3-4 Stunden

Visualisierung des Preisverlaufs für Spiele über die Zeit. Zeigt historische Preise, Sale-Trends und beste Kaufzeitpunkte.

Warum dieses Feature?

- **Nutzerwert:** Hilft bei Kaufentscheidungen ("Wann war der letzte Sale?")
- **SEO:** Long-Tail Keywords wie "hollow knight preisverlauf"
- **Engagement:** Nutzer bleiben länger auf der Seite
- **Daten vorhanden:** `price_snapshots` Tabelle existiert bereits!

Features

1. **Preisverlauf-Chart** auf Game-Detail-Seite
    - Line Chart mit Zeitachse (x) und Preis (y)
    - Markierungen für Sale-Perioden
    - Tooltip mit exaktem Preis und Datum

2. **Preis-Alert Historie**
    - Zeigt wann Push-Notifications gesendet wurden
    - "Du wurdest am 15.12. über diesen Sale informiert"
3. **"Beste Kaufzeit" Badge**
    - Berechnet historisches Preis-Tief
    - "Niedrigster Preis: 9,99€ (vor 2 Monaten)"

Technische Umsetzung

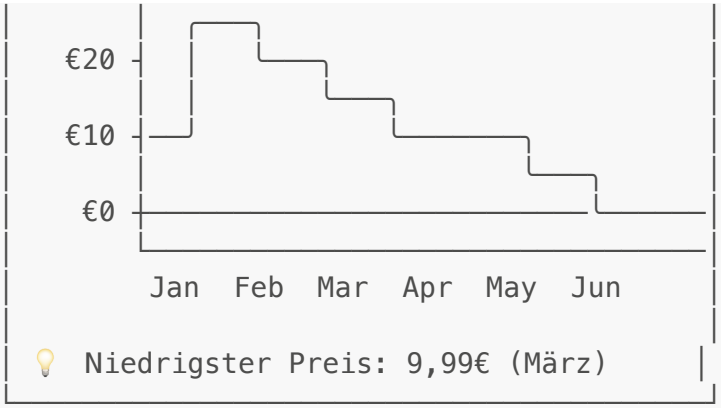
```
// src/lib/queries.ts

export async function getPriceHistory(gameId: string, store: string =
'steam') {
  return db
    .select({
      priceFinal: priceSnapshots.priceFinal,
      priceInitial: priceSnapshots.priceInitial,
      discountPercent: priceSnapshots.discountPercent,
      isOnSale: priceSnapshots.isOnSale,
      fetchedAt: priceSnapshots.fetchedAt,
    })
    .from(priceSnapshots)
    .where(
      and(
        eq(priceSnapshots.gameId, gameId),
        eq(priceSnapshots.store, store)
      )
    )
    .orderBy(desc(priceSnapshots.fetchedAt))
    .limit(365); // Letztes Jahr
}
```

Chart-Komponente:

- Plotly.js für interaktive Charts (bereits in user_rules erwähnt)
- Oder Chart.js/Recharts für einfachere Lösung
- Responsive Design

UI/UX

```
┌─────────────────────────────────────────┐
│ Preisverlauf (Steam)                    │
├─────────────────────────────────────────┤
│                                         │
│   €30 ┤                                 │
```



Benötigte Dateien

Aktion	Datei	Beschreibung
Ändern	`src/lib/queries.ts`	`getPriceHistory()` Funktion
Neu	`src/components/PriceChart.tsx`	Chart-Komponente
Ändern	`src/app/[lang]/games/[slug]/page.tsx`	Chart einbinden
Ändern	`src/dictionaries/*.json`	Übersetzungen

---

3. Erweiterte Filter auf Games-Seite

Status: Idee
Geschätzter Aufwand: ~2-3 Stunden

Aktuell gibt es nur Sortierung. Filter nach Tags, Score-Bereich, Plattform, Preis würden die Nutzbarkeit deutlich verbessern.

Filter-Optionen

1. **Tags** (Multi-Select)
   - "Roguelike", "Metroidvania", "Puzzle", etc.
   - Checkboxen oder Tag-Chips
2. **Score-Bereich** (Range Slider)
   - Min: 0-10, Max: 0-10
   - Oder vordefinierte Bereiche: "7+", "8+", "9+"
3. **Plattform** (Multi-Select)
   - PC, Switch, PlayStation, Xbox, etc.
4. **Preis** (Range)

- "Unter 10€", "10-20€", "20-50€", "50€+"
- Oder "Nur im Sale"

5. **Release-Datum**
   - "Letztes Jahr", "Letzte 2 Jahre", "2024", etc.

URL-Struktur

```
/games?tags=roguelike,metroidvania&minScore=7&platform=pc&maxPrice=2000
```

Vorteile:

- Shareable URLs
- Browser-Back/Forward funktioniert
- SEO-freundlich (kann später indexiert werden)

Technische Umsetzung

```
// src/lib/queries.ts

export async function getFilteredReviews(filters: {
  tags?: string[];
  minScore?: number;
  maxScore?: number;
  platforms?: string[];
  maxPrice?: number;
  minPrice?: number;
  releasedAfter?: Date;
  orderBy?: SortOrder;
}): Promise<ReviewListItem[]>
```

Filter-Komponente:

- Client-Side Filter-UI (`FilterPanel.tsx`)
- Server-Side Query-Parameter Parsing
- URL-State Management

UI/UX

```
┌──────────────────────────────────────────┐
│ Alle Spiele (127)          [Sortieren ▼] │
├──────────────────────────────────────────┤
│ Filter:                                  │
│ ☑ Roguelike  □ Metroidvania  □ Puzzle    │
```

```
┌──────────────────────────────────────────┐
│                                          │
│ Score: [━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━] │
│        7.0                    10.0       │
│                                          │
│ Plattform: ☑ PC  ☐ Switch  ☐ PlayStation │
│                                          │
│ Preis: [Unter 10€] [10–20€] [20€+]       │
│                                          │
│ [Filter zurücksetzen]                    │
└──────────────────────────────────────────┘
```

---

4. Vergleichsfunktion (Compare Games)

Status: Idee
Geschätzter Aufwand: ~4-5 Stunden

Nutzer können 2-3 Spiele nebeneinander vergleichen (Score, Preis, Tags, Features).

Warum dieses Feature?

- **Kaufentscheidung:** "Soll ich Game A oder Game B kaufen?"
- **Engagement:** Nutzer bleiben länger auf der Seite
- **Social Sharing:** Vergleichs-URLs werden geteilt

Features

1. **"Vergleichen" Button** auf Game-Cards
    - Maximal 3 Spiele gleichzeitig
    - Badge zeigt Anzahl im Vergleich
2. **Vergleichs-Seite** `/compare?games=slug1,slug2,slug3`
    - Side-by-Side Vergleich
    - Score, Preis, Tags, Pros/Cons, Release-Datum
3. **LocalStorage** für Persistenz
    - Vergleich bleibt erhalten beim Navigieren
    - Oder URL-basiert (shareable)

UI/UX

```
┌──────────────────────────────────────────────────────┐
│ Spiele vergleichen                                   │
├──────────────────┬──────────────────┬────────────────┤
│ Hollow Knight    │ Dead Cells       │ [Spiel +]      │
```

```
┌──────────────────────┬──────────────────────┬──────────────┐
│                      │                      │              │
│ Score: 9.5 ⭐        │ Score: 9.0 ⭐        │              │
│ Preis: 14,99€        │ Preis: 24,99€        │              │
│                      │                      │              │
│ Tags:                │ Tags:                │              │
│ • Metroidvania       │ • Roguelike          │              │
│ • Indie              │ • Action             │              │
│                      │                      │              │
│ Pros:                │ Pros:                │              │
│ • Fantastisches...   │ • Perfekte...        │              │
│                      │                      │              │
│ [Entfernen]          │ [Entfernen]          │              │
└──────────────────────┴──────────────────────┴──────────────┘
```

Technische Umsetzung

Client-Side State:

```
// src/lib/compare.ts
export function addToCompare(slug: string): void
export function removeFromCompare(slug: string): void
export function getCompareList(): string[]
export function clearCompare(): void
```

Server-Side Query:

```
// src/lib/queries.ts
export async function getGamesBySlugs(slugs: string[]):
Promise<GameDetail[]>
```

Vergleichs-Seite:

- `/[lang]/compare/page.tsx` - Lädt Spiele basierend auf URL-Params oder LocalStorage

---

5. Newsletter / Email-Benachrichtigungen

Status: Idee
Geschätzter Aufwand: ~5-6 Stunden

Alternative zu Push-Notifications: Email-Benachrichtigungen für neue Reviews oder Sales.

Warum dieses Feature?

- **Reichweite:** Nicht alle Nutzer aktivieren Push-Notifications
- **Engagement:** Email hat höhere Öffnungsrate bei wichtigen Updates

- **Newsletter:** Wöchentliche Zusammenfassung "Top 5 Reviews dieser Woche"

Features

1. **Email-Subscription** auf Wishlist-Seite
   - "Erhalte Email-Benachrichtigungen für deine Wishlist"
   - Double-Opt-In
2. **Email-Templates**
   - Neue Review veröffentlicht
   - Sale-Alert für Wishlist-Spiele
   - Wöchentlicher Newsletter
3. **Unsubscribe-Seite**
   - `/unsubscribe?token=...`
   - Ein-Klick-Deaktivierung

Technische Umsetzung

Email-Service:

- Resend.com (einfach, günstig)
- Oder SendGrid/Mailgun
- Oder AWS SES (kostenlos für kleine Volumes)

Schema-Erweiterung:

```
export const emailSubscriptions = pgTable('email_subscriptions', {
  id: uuid('id').defaultRandom().primaryKey(),
  email: text('email').notNull().unique(),
  isActive: boolean('is_active').default(true),
  wishlistAlerts: boolean('wishlist_alerts').default(true),
  weeklyNewsletter: boolean('weekly_newsletter').default(false),
  unsubscribeToken: text('unsubscribe_token').notNull().unique(),
  createdAt: timestamp('created_at', { withTimezone: true }).defaultNow(),
});
```

API-Route:

- `POST /api/email/subscribe` - Neue Subscription
- `POST /api/email/unsubscribe` - Deaktivieren
- `GET /api/cron/send-emails` - Cron-Job für Versand

---

6. "Zuletzt angesehen" (Recently Viewed)

Status: Idee
Geschätzter Aufwand: ~1-2 Stunden

Einfaches Feature: Zeigt die zuletzt besuchten Spiele-Seiten.

Features

1. **LocalStorage-basiert**
   - Speichert Slugs der letzten 5-10 besuchten Spiele
   - Keine Server-Logik nötig
2. **UI-Integration**
   - Dropdown im Header: "Zuletzt angesehen"
   - Oder Sektion auf Homepage/Wishlist-Seite
3. **Optional: Server-Side**
   - Falls User-Accounts später kommen
   - Dann in DB speichern

Technische Umsetzung

```
// src/lib/recently-viewed.ts

const STORAGE_KEY = 'recently-viewed';
const MAX_ITEMS = 10;

export function addRecentlyViewed(slug: string): void {
  const current = getRecentlyViewed();
  const updated = [slug, ...current.filter(s => s !== slug)].slice(0,
MAX_ITEMS);
  localStorage.setItem(STORAGE_KEY, JSON.stringify(updated));
}

export function getRecentlyViewed(): string[] {
  if (typeof window === 'undefined') return [];
  const stored = localStorage.getItem(STORAGE_KEY);
  return stored ? JSON.parse(stored) : [];
}
```

Komponente:

- `RecentlyViewed.tsx` - Zeigt Liste mit Links zu Spielen
- Lädt Game-Daten via `getGamesBySlugs()`

---

7. Social Sharing Buttons

Status: Idee
Geschätzter Aufwand: ~1 Stunde

Share-Buttons für Twitter/X, Facebook, Reddit auf Game-Detail-Seiten.

Features

1. **Share-Buttons** auf Game-Detail-Seite
    - Twitter/X: "Check out {Game} - Score: {Score} ⭐ "
    - Facebook: Open Graph wird automatisch genutzt
    - Reddit: Link zu r/gaming oder r/IndieGaming
    - Copy-Link Button
2. **Share-Image**
    - Generiertes Bild mit Game-Cover, Score, Titel
    - Oder einfach Hero-Image nutzen

Technische Umsetzung

Komponente:

```
// src/components/ShareButtons.tsx

type ShareButtonsProps = {
  title: string;
  slug: string;
  score?: number;
  locale: Locale;
};
```

Share-URLs:

- Twitter: `https://twitter.com/intent/tweet?text=...&url=...`
- Facebook: `https://www.facebook.com/sharer/sharer.php?u=...`
- Reddit: `https://reddit.com/submit?url=...&title=...`

---

8. Erweiterte Suche mit Autocomplete

Status: Idee
Geschätzter Aufwand: ~2-3 Stunden

Aktuell gibt es eine Search-Route. Erweitern um Autocomplete-Dropdown während des Tippens.

Features

1. **Live-Search** während des Tippens
    - Debounced API-Calls (300ms)
    - Dropdown mit Vorschlägen
2. **Suche in:**
    - Spiel-Titel
    - Developer/Publisher
    - Tags
3. **Keyboard-Navigation**
    - Arrow-Keys zum Navigieren
    - Enter zum Auswählen

Technische Umsetzung

API-Route erweitern:

```
// src/app/api/search/route.ts

// Query-Parameter: ?q=...&limit=5
// Gibt erste 5 Ergebnisse zurück für Autocomplete
```

Client-Komponente:

- `SearchBar.tsx` erweitern
- Debounce mit `useDebouncedCallback`
- Dropdown mit `SearchResults.tsx`

---

9. RSS Feed erweitern

Status: Idee
Geschätzter Aufwand: ~1-2 Stunden

Aktuell gibt es `/rss.xml`. Erweitern um mehrere Feeds:

Feed-Varianten

1. **Hauptfeed** (bereits vorhanden)
    - Alle neuen Reviews
2. **Tag-spezifische Feeds**
    - `/rss.xml?tag=roguelike`
    - `/rss.xml?tag=metroidvania`

3. **Score-Filter**
    - `/rss.xml?minScore=8`
    - Nur Reviews mit Score 8+
4. **Wöchentlicher Digest**
    - `/rss.xml?type=weekly`
    - Zusammenfassung der Top-Reviews der Woche

Technische Umsetzung

RSS-Route erweitern:

```
// src/app/rss.xml/route.ts

// Query-Parameter parsen
// Filter anwenden
// Feed generieren
```

SEO:

- `<link rel="alternate" type="application/rss+xml">` Tags auf relevanten Seiten
- Sitemap-Einträge für Feed-URLs

---

10. User-Bewertungen (Optional, später)

Status: Zukunftsvision
Geschätzter Aufwand: ~8-10 Stunden

Nutzer können eigene Bewertungen/Reviews zu Spielen abgeben (komplementär zu den kuratierten Reviews).

Features

1. **User-Accounts** (optional)
    - Oder anonyme Bewertungen mit Cookie/Device-ID
2. **Bewertungs-Formular**
    - Score (1-10)
    - Kurzer Kommentar
    - Pros/Cons
3. **Aggregierte Anzeige**
    - "Community Score: 8.5 (127 Bewertungen)"

    - Neben dem kuratierten Score

Warum optional?

- **Komplexität:** Benötigt User-Management, Moderation
- **Qualität:** Kuratierte Reviews sind wertvoller
- **Später:** Kann später hinzugefügt werden wenn Traffic vorhanden

---

Priorisierung

Feature	Aufwand	Nutzen	Priorität
Kuratoren-Listen	4-5h	⭐⭐⭐⭐⭐	**Hoch**
Erweiterte Filter	2-3h	⭐⭐⭐⭐	**Hoch**
Preisverlauf	3-4h	⭐⭐⭐⭐	**Mittel**
Vergleichsfunktion	4-5h	⭐⭐⭐	**Mittel**
Zuletzt angesehen	1-2h	⭐⭐⭐	**Mittel**
Social Sharing	1h	⭐⭐	**Niedrig**
Erweiterte Suche	2-3h	⭐⭐⭐	**Mittel**
RSS erweitern	1-2h	⭐⭐	**Niedrig**
Newsletter	5-6h	⭐⭐⭐⭐	**Mittel**
User-Bewertungen	8-10h	⭐⭐⭐	**Später**

Weitere Ideen

- **Dark Mode Toggle** (falls noch nicht vorhanden)
- **PWA-Features erweitern** (Offline-Modus, Install-Prompt)
- **Game-Trailer Integration** (YouTube/Vimeo Embed)
- **"Ähnliche Spiele" Algorithmus verbessern** (ML-basiert)
- **Multi-Store Preisvergleich** (Steam, Epic, GOG)
- **Review-Kommentare** (Diskussionen zu Reviews)
- **Export-Funktion** (Wishlist als CSV/JSON exportieren)