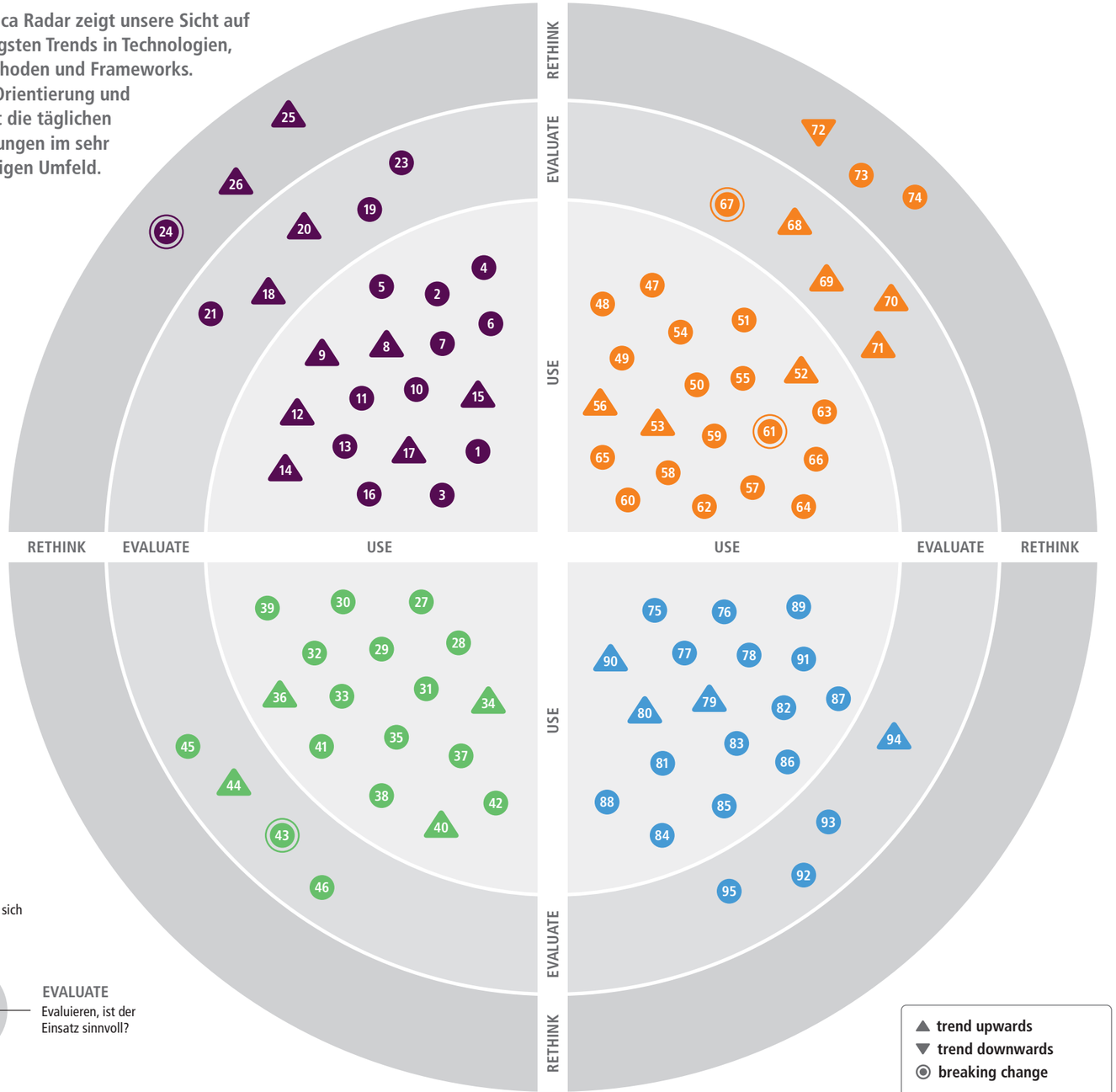
bbv
MAKING VISIONS WORK.

TECHNICA RADAR – VOL. 4/2024

Der Technica Radar zeigt unsere Sicht auf die wichtigsten Trends in Technologien, Tools, Methoden und Frameworks. Er bietet Orientierung und erleichtert die täglichen Entscheidungen im sehr schnelllebigen Umfeld.



Concept or theme

USE
1 Advanced Analytics / Machine Learning
2 Clean Architecture / Ports & Adapters / Hexagonal Architecture / Onion Architecture
3 Cloud Native
4 Design system
5 DevOps
6 Domain Driven Design
7 Edge Computing
8 Emergent Architecture ▲
9 Evolutionary Architecture ▲
10 Functional Programming
11 Hybrid Cloud
12 Inclusive Design ▲
13 Internet of Things
14 Product Life Cycle Based Development ▲
15 Security Engineering ▲
16 Serverless Architecture
17 Wertstromorganisation ▲

EVALUATE
18 AI Ethics ▲
19 CUPID
20 Ethics in Product Development ▲
21 Platform economy
22 Sustainable Software Engineering
23 WebAssembly

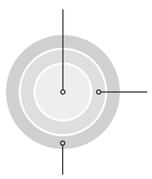
RETHINK
24 Agile ◉
25 Datenstrategie ▲
26 Democratizing Programming ▲

Tools

USE
27 Cloud Computing
28 Cumulocity IoT
29 Diagram as Code
30 Docker
31 Dynamic Application Security Testing (DAST)
32 ESLint
33 Figma
34 Infrastructure as Code ▲
35 Kubernetes
36 Large Language Models ▲
37 Miro
38 Prettier
39 Software as a Service (SaaS)
40 Software Composition Analysis incl. SBOM ▲
41 Static Application Security Testing (SAST)
42 Vite

EVALUATE
43 AI im O365 / Windows ◉
44 GitHub Copilot ▲
45 No-Code / Low-Code
46 WebAssembly

USE
Anwenden, hat sich bewährt.



EVALUATE
Evaluieren, ist der Einsatz sinnvoll?

RETHINK
Technologien und Rahmenbedingungen ändern sich und müssen manchmal neu betrachtet werden.

▲ trend upwards
▼ trend downwards
◉ breaking change

Beobachtete Trends im Markt und in Projekten. Die Analyse unserer Experten dazu auf der Folgeseite.

Method or technique

USE
47 Acceptance Test Driven Development (ATDD) / Behavior Driven Development (BDD)
48 arc42 [Documentation]
49 Continuous Delivery
50 Continuous Refactorings / RefactorMercilessly
51 Cost of Delay
52 Cyber / Security Resilience ▲
53 Decision Log ▲
54 Design system
55 Design Thinking
56 Ethical OS ▲
57 Event Storming
58 Liberating Structures
59 Pair Programming
60 Risk management
61 Security by design & Defense in depth ◉
62 Service Design
63 Systems Thinking
64 Test Driven Development (TDD)
65 User Research
66 User-Centered-Design

EVALUATE
67 AI in Software Engineering ◉
68 Explainable AI (xAI) ▲
69 FAIR Principle ▲
70 Prompt Engineering ▲
71 Retrieval Augmented Generation (RAG) ▲

RETHINK
72 Effort Estimation ▼
73 Mob programming
74 SAFe – Scaled Agile Framework

Libraries, Frameworks, Programming Languages

USE
75 .NET
76 Angular
77 Azure IoT Edge
78 C++
79 F# ▲
80 GPT-4 ▲
81 gRPC
82 Java
83 Jest
84 Kotlin
85 Lit
86 OpenTelemetry
87 Playwright
88 Python
89 React
90 Rust ▲
91 Spring Framework

EVALUATE
92 Distributed Application Runtime (Dapr)
93 Go Programming Language
94 Open Source LLMs ▲
95 R

Vol. 4 – Published: March 2024
This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Die Analyse der wichtigsten Trends

▲ trend upwards ▼ trend downwards ◉ breaking change

Concept or theme

USE

8 Emergent Architecture ▲

In einer komplexen Umgebung ist es unmöglich, eine Architektur endgültig zu entwerfen. Dennoch sollte die Architektur die aktuellen architekturellen Probleme lösen. Ein Lösungsansatz: Emergent Architecture. Statt die Architektur vor der Entwicklung final zu entwerfen, wird sie mittels Modularisierung und Clean Architecture fortlaufend weiterentwickelt.

9 Evolutionary Architecture ▲

In einer komplexen Entwicklungsumgebung muss Software-Architektur flexibel bleiben. Weil sich die eingesetzten Technologien, Cloud-Dienste und Business-Prozesse laufend verändern, wird die Architektur teilweise ersetzt oder verbessert.

12 Inclusive Design ▲

Inclusive Design verhindert, dass gewisse Nutzer von der Verwendung einer digitalen Lösung ausgeschlossen sind. Inclusive Design geht weiter als Barrierefreiheit, da unter anderem auch kulturelle und demografische Aspekte betrachtet werden.

14 Product Life Cycle Based Development ▲

Wie in der klassischen Industrie muss sich die Software-Entwicklungsindustrie über den ganzen Software-Produktlebenszyklus Gedanken machen. Dies kann die Gesamtheit der Kosten einer Investition, die über ihren kompletten Lebenszyklus anfallen (total cost of ownership), verkleinern. Im Vergleich zu stabilen Projektteams führen stabile Produktteams zu weniger Spannungen. Weitere Vorteile sind eine kontinuierliche, stabile Weiterentwicklung, sich evolvierende Systeme und stabileres Domänenwissen.

15 Security Engineering ▲

Security Engineering ist das technische Wissen, wie ein Secure Software Development Lifecycle umgesetzt wird. Know-how rund um Security in der Entwicklung und das Bewusstsein für Sicherheitsgrundlagen, sichere Codier-Praktiken, Schutz vor OWASP-Top-10 oder CVEs sind heutzutage notwendig. Dies insbesondere mit Blick auf neue Regulatorien, die demnächst in Kraft treten (NIS-2, RED, CRA).

17 Wertstrom-Organisation ▲

Kern der Wertstrom-Organisation ist, das Unternehmen auf die Wertschöpfung des Kunden auszurichten und den Kunden ins Zentrum der Aktivitäten zu stellen. Die Organisation soll dabei bestmöglich die Mitarbeitenden unterstützen, die Mehrwert für den Kunden generieren.

EVALUATE

18 AI Ethics ▲

Neben den OECD-Grundsätzen für künstliche Intelligenz aus dem Jahr 2019 und den Guidelines on AI for the Confederation von 2020 sehen wir häufiger AI Ethics von Firmen erscheinen, die sich stark mit dem KI-Thema und KI-Produkten auseinandersetzen oder sogar eigene KI erschaffen. So hat z. B. Microsoft ethische Prinzipien für die KI entwickelt. Die meisten dieser Grundätze überschneiden sich mit den gängigen Ethik-Prinzipien, die in der Produktentwicklung anzutreffen sind.

20 Ethics in Product Development ▲

In einer von Technologie getriebenen Produktentwicklung besteht das Risiko, dass ethische Aspekte bei der Produktentwicklung zu kurz kommen. Die wichtigsten Aspekte sind Privacy, die Beachtung von Bias, die Vermeidung von Suchtmechanismen, Accessibility, Nachvollziehbarkeit und Auswirkungen auf die Umwelt. Um diese Aspekte in die Produktentwicklung zu integrieren, braucht es ein ethisches Framework und Prozesse, die eine ethische Betrachtung bei der Produktentwicklung sicherstellen.

RETHINK

24 Agile ◉

Agile ist tot, lang lebe Agile. Kaum ein Begriff ist so oft und so unterschiedlich interpretiert wie das Wort agile. Ursprung des agilen Denkens ist das agile Manifesto und die 12 dazugehörigen Prinzipien. Es tut der Diskussion gut, sich wieder auf das Manifesto zu besinnen, und vieles, was damit vermischt wird, davon zu trennen. Ein Beispiel ist Scrum: Scrum ist ein Framework, das ein Team mit agilem Mindset unterstützt. Nicht mehr, nicht weniger. Die buchstabengetreue Umsetzung von Scrum muss nicht zu agilen Teams führen, auch wenn der Scrum-Kult gerne etwas anderes sagt. Scrum allein, ohne Teamtraining, ohne Lean-Gedanke und ohne Coaching oder Mentoring zur Selbstorganisation, macht nicht agil. Gleiches gilt auch für SAFe. Unternehmen müssen Teamführung zudem neu denken: weg vom direkten Führen hin zum Setzen von Rahmenbedingungen, die auf die Fähigkeiten der Teams angepasst sind und kontinuierlich weiterentwickelt werden.

25 Datenstrategie ▲

Die Datenstrategie ist für den Erfolg eines Unternehmens entscheidend. Wichtig ist dabei, welche Mitarbeitenden welche Daten verwenden dürfen, wie die Datenqualität sichergestellt wird und den Zweck aller erfassten Daten zu definieren. Dies ist die Grundlage für Prozessverbesserungen und Prozessautomatisierung, wie auch für das Design digitaler Services für Kunden.

26 Democratizing Programming ▲

Democratizing Programming und Low Code sind Möglichkeiten, ohne Studium oder Ausbildung in Informatik die Businesslogik einer Applikation in Teilen zu automatisieren oder gesamte Applikationen zu entwickeln. Democratizing Programming ersetzt keine klassische IT-Entwicklung und sollte zusammen mit der Softwareentwicklung im Unternehmen eingeführt werden, um keine falschen Erwartungen zu wecken und Missverständnisse zu vermeiden. Dieses Konzept kann dabei unterstützen, Softwareentwicklung und Business einander näherzubringen.

Tools

USE

34 Infrastructure as Code ▲

Infrastrukturen werden komplizierter. Um die Qualität aufrechtzuerhalten und Migrationen zu automatisieren, werden Infrastrukturen in maschinenlesbarem Format definiert und versioniert. Ein Trade-off zwischen manuell und IaC sollte bedacht werden. Der IaC-Lösung sollte der Vorzug gegeben werden.

36 Large Language Models ▲

Nicht neu, aber mit dem Release von GPT-3 hat die Technologie an Fahrt aufgenommen und einen überraschend starken Einfluss auf die Industrie. Im Unterschied zu vorangegangenen Lösungen ist es keine eindimensionale Lösung, die auf einem Spezialgebiet verwendet wird, sondern eine flexible Lösung, die in mehreren Einsatzgebieten verwendet werden kann. Der Einfluss kann daran gemessen werden, dass nach dem Erscheinen von GPT-3 einige der grossen Digitalunternehmen rasch ähnliche Lösungen, wie Google Bard, auf den Markt gebracht haben. LLMs sind im Alltag angekommen und werden uns künftig begleiten.

40 Software Composition Analysis ▲

Immer mehr Funktionalitäten einer Software werden durch Dritt-Komponenten gelöst. Diese Komponenten, egal ob Open Source oder nicht, bergen ein inhärentes Risiko bezüglich Security. Mit der Software Composition Analysis wird anhand einer Software Bill of Material (SBOM) eine Bestandsanalyse der verwendeten Software-Komponenten gemacht. Dabei werden diese getrackt und gegen bekannte Schwachstellen (CVE) geprüft. Somit wird ein proaktives Monitoring gegen bekannte Schwachstellen aufgebaut. Nebenbei prüft die Methode, ob die Lizenzierung der eingesetzten Komponenten konform ist.

43 AI im O365/Windows ◉

Mit der Integration von Copilot und somit von KI direkt in Windows und den O365-Tools wird sich das Arbeiten in Word, Excel und PowerPoint stark verändern. Wir sehen gerade in diesem Umfeld eine mögliche Produktivitätssteigerung.

44 GitHub Copilot ▲

GitHub Copilot kann Code in verschiedenen Programmiersprachen generieren und vervollständigen, basierend auf den Kommentaren und dem bereits geschriebenen Code des Benutzers. Es funktioniert als Plug-in für verschiedene Entwicklungsumgebungen, hilft Entwicklern, indem es Vorschläge für ganze Zeilen oder Blöcke von Code macht oder Bugs identifiziert. Der Copilot wird aber auch auf die GitHub Repositories ausgeweitet, wo dieser das Suchen im Repository unterstützt oder automatisch Commit-Messages und Codereviews schreibt.

Method or technique

USE

53 Decision Log ▲

Ein Logbuch der Entscheidungen ist ein einfaches Mittel, einen Überblick über diese zu behalten. Es sollten alle relevanten Entscheidungen dokumentiert werden, so können Gründe, Alternativen, Kontext und die Entscheidung im Nachhinein nachvollzogen werden. Solche Logbücher sind in der Architektur als Architecture Decision Log bekannt, könnten jedoch auch in Design-Systemen, Projekten, Produkten oder Team-Setups Einzug finden.

56 Ethical OS ▲

Ethical OS ist ein Framework, das Start-ups und Produktentwicklern hilft, die moralischen Herausforderungen ihrer Produkte zu bewältigen. Es befasst sich zum Beispiel mit Features, die ein Produkt in ein moralisch fragwürdiges Licht rücken könnten. Beispiel: Das Feature zur Ortung eines verlorenen Mobiltelefons wird zur Überwachung einer Person eingesetzt.

61 Security by design & Defense in depth ◉

Security und ein Secure Software Development Lifecycle sind wichtiger denn je. Bei Security by Design fliessen Security-Überlegungen von Anfang an in jede Phase des Entwicklungsprozesses ein. Zum Beispiel hilft kontinuierliches Threat Modeling, Schwachstellen bereits in der Architektur- und Design-Phase des Systems zu erkennen. Das Grundprinzip eines Secure SW Development Lifecycles ist das Shift-Left-Prinzip. Dadurch werden Schwachstellen frühzeitig erkannt und sie können kostengünstiger beseitigt werden. Weiter sind der Einsatz von Analysetools zu statischen und dynamischen Analysen sowie Vulnerability-Scans sinnvoll. Mit Defense-in-Depth wird ein mehrschichtiges, redundantes, überwachtes Sicherheitssystem verfolgt, das im Ernstfall durch die festgelegten Massnahmen keine grösseren Schäden nimmt.

EVALUATE

67 AI in Software Engineering ◉

AI spielt im Software Engineering eine vielseitige Rolle. AI unterstützt bei der Analyse und Priorisierung von Anforderungen, indem sie aus grossen Mengen von Benutzerfeedbacks Muster erkennt und wertvolle Einsichten liefert. Sie ermöglicht zudem die Automatisierung von Testprozessen, die effiziente Erkennung von Softwarefehlern und die Optimierung der Testabdeckung. Im Projektmanagement unterstützt AI durch fortschrittliche Analytik und Vorhersagemodelle, um Risiken zu minimieren, Ressourcen effektiv zu verwalten und Entscheidungsprozesse zu verbessern.

68 Explainable AI (xAI) ▲

Explainable AI, also erklärbare, nachvollziehbare Künstliche Intelligenz, beschreibt eine Sammlung von Methoden und Prozessen, um den Zweck, die Auswirkungen und potenzielle Bias eines Modells zu beschreiben. Dies soll das Vertrauen in KI-Lösungen stärken, indem die Ergebnisse einer solchen Lösung nachvollziehbar sind.

69 FAIR Principle ▲

FAIR steht für Findable, Accessible, Interoperable und Reusable (auffindbar, zugreifbar, interoperabel und wiederverwendbar). FAIR ist ein Prinzip des Datenmanagements. Es vereinfacht die Arbeit mit Daten und ist die Grundlage für die systematische Verwendung datengetriebener Lösungen wie beispielsweise AI.

70 Prompt Engineering ▲

Der Hype um Generative AI und Conversational User Interfaces hat dazu geführt, dass Prompt Engineering ein wichtiger Skill für Anwender geworden ist. Linguistische Kenntnisse oder die Kenntnis von Fachbegriffen sind nicht notwendig, können jedoch helfen, die Qualität des Outputs zu verbessern.

71 Retrieval Augmented Generation (RAG) ▲

Dieses Konzept kombiniert die Informationssuche (Retrieval) mit der Erzeugung von Text (Generation). Dabei sucht das System zunächst nach relevanten Informationen in einer grossen Datenbank, um kontextbezogene und spezifischere Antworten zu liefern, als es durch alleinige Textgenerierung durch ein LLM möglich wäre, ohne das LLM spezifisch zu trainieren. Zur effizienten Suche von relevanten Informationen werden oft sogenannte Vektordatenbanken eingesetzt.

RETHINK

72 Effort Estimation ▼

In einem komplexen Umfeld ist vieles nicht vorhersehbar. Daher sind zuverlässige Schätzungen über einen längeren Zeitraum unmöglich. Anstelle von vermeintlich genauen Plänen und der Frage «Wie aufwendig ist das?» sollten Teams den Fokus auf «Was gewinnen wir?» legen und kontinuierlich an Produktinkrementen arbeiten, die den bestmöglichen Wert generieren.

Libraries, Frameworks, Programming Languages

USE

79 F# ▲

Die prägnante, robuste «functional-first»-Sprache F# hat eine viel kleinere Sprachspezifikation als C# und ist damit einfacher zu lernen und einzusetzen. Trotzdem ist sie als Vorreiter oft die Quelle für die neueren Sprachfeatures von C#, auch wenn sie nicht immer gleichwertig umgesetzt werden. F# kann für die Entwicklung eines Frontends, eines Backends oder zur Infrastruktur verwendet werden.

80 GPT-4 ▲

GPT-4 ist die aktuelle Version von Open AIs multimodalem Large Language Modell (LLM). Es wird unter anderem von ChatGPT und Microsofts Copilot verwendet. Copilot ist nach einem Update Bestandteil von Windows 11 und so einer breiten Anwenderbasis zugänglich. Eine tiefere Integration in Windows 12 wird erwartet.

EVALUATE

94 Open Source LLMs ▲

In der schnelllebigen Welt der Künstlichen Intelligenz haben sich Open Source Large Language Models (LLMs) als bedeutende Akteure etabliert. Diese Modelle sind oft auf spezifische Fähigkeiten hin optimiert und sind in diesen Bereichen häufig State of the Art (SOTA). Auf dem Huggingface Open LLM Leaderboard wird ihre Leistungsfähigkeit verglichen. Ein signifikanter Vorteil dieser Modelle ist ihre Fähigkeit, auf eigener Infrastruktur betrieben zu werden. Die Grössen dieser Modelle variieren, wobei die gängigsten Konfigurationen 7, 14 und 70 Milliarden Parameter umfassen. Mit dem Aufkommen von Mixture-of-Experts-Modellen (MoE) Ende 2023 wurden weitere Fortschritte erzielt, insbesondere in der Anwendung als Chatbots, wie das Chatbot Arena-Leaderboard zeigt. Obwohl derzeit proprietäre Modelle populärer sind, können Open-Source-Modelle richtig eingesetzt eine überzeugende Alternative darstellen, besonders hinsichtlich ihrer Anpassungsfähigkeit und Zugänglichkeit für eine breite Nutzerschaft.

Dieser Radar kann Spuren von AI generierten Texten enthalten.

Legal Disclaimer: While we have made every attempt to ensure that the information in this publication has been obtained from reliable sources, bbv Software Services AG (bbv) is not responsible for any errors or omissions, or for the results obtained from the use of this information. All information is provided with no guarantee of completeness or accuracy, and without warranty of any kind. In no event will bbv or its employees therefore be liable to you or anyone else for any decision made or action taken in reliance on the information in this publication. The information in this publication should not be used as a substitute for consultation with professional bbv advisors. Before making any decision or taking any action, you should consult a bbv professional. The names of actual companies and products mentioned in this publication may be the trademarks of their respective owners.

Vol. 4 – Published: March 2024
This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.