

Dominik Meyer

Software Engineering

Entwurfsmuster

WEITER WISSEN →



Ziel

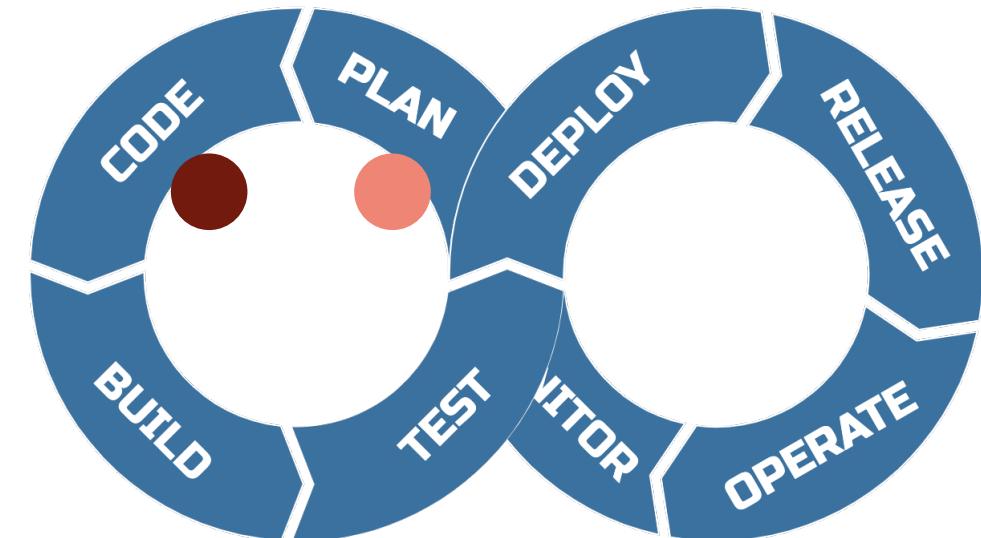
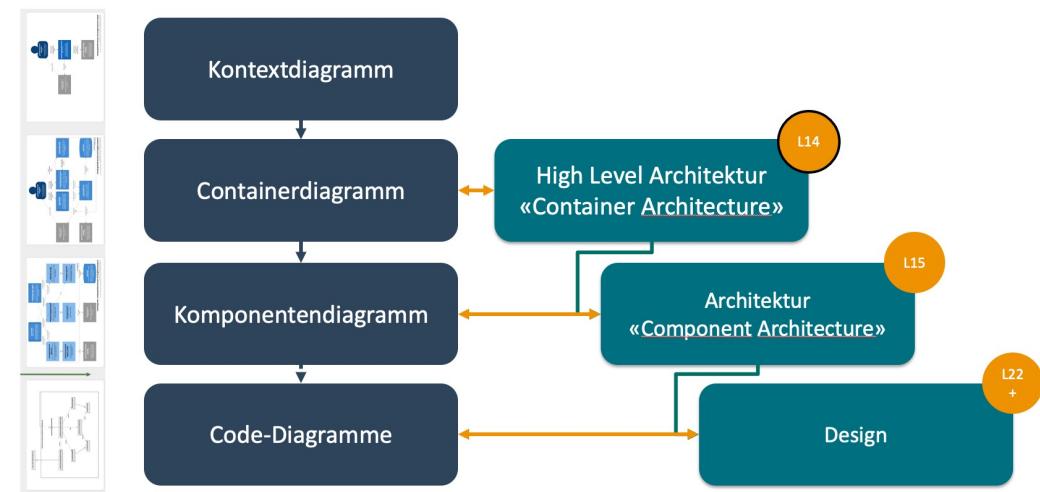
Nach der Lektion haben beschreiben die Studierenden (und wenden an) die Funktionen `git add/commit/push/checkout` (Feedback).

Die Studierenden können nachvollziehbar aufzeichnen, wie aus C4 Modellen und OOP ein Software Design erstellt wird.

Entwurfsmuster

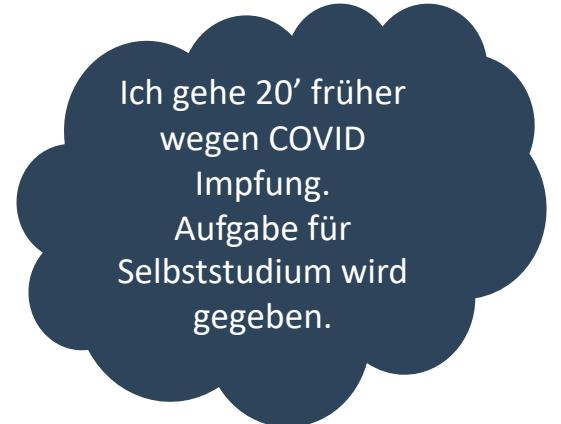
- Architektur
- Container für Tests und Delivery
- Unit -& Integrationstests
- Github Actions zur Automation
- Applikation gem.
«Design Considerations»
- Entwurfsmuster

Unsere Flughöhen



Agenda

- Ziel
- Standortbestimmung
- Transferaufgaben
 - Architektur
 - Integration Testing
- Repetition Git (Feedback)
- Repetition Controller
- Entwurfsmuster
 - SOLID
 - GRASP
- Zielkontrolle

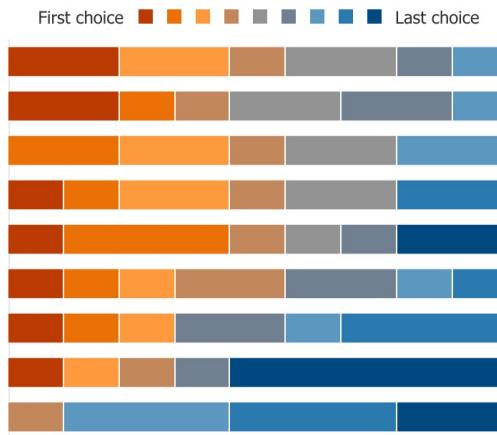


Ich gehe 20' früher wegen COVID Impfung.
Aufgabe für Selbststudium wird gegeben.

Standortbestimmung

Rank Options

- 1 SOLID Design Prinzipien verst...
- 2 Entwurfsmuster (C4 Modelle i...
- 3 Clean Code und Refactoring
- 4 Daten persistieren (mit einer ...
- 5 Kontinuierlich Deployen/Relea...
- 6 GRAP Design Prinzipien verste...
- 7 Applikationen mit Logs debug...
- 8 Applikationen mit Telemetrie ...
- 9 Applikationen mit Swagger un...



Ich fände es interessant wenn wir die Architekturformen mit Microservices genauer anschauen und hier vor allem den Zusammenschluss dieser beleuchten - sprich, wie stelle ich sicher, dass alle laufen, wie tracke ich deren Status... (ich rüste meine Services mit REST aus und kann deren Status so überwachen... Die Frage ist obs hier andere Strategien gäbe) Ebenfalls wäre es gut, wenn wir bezüglich Realisierung der Sicherheitsmassnahmen einen Einschub machen könnten (Ich denke da an strategische Themen, wie warum sollten aten von Logik getrennt werden usw..)

MS: 😢⏰

Ich verstehe das man einem Lehrplan folgen muss um möglichst breit Themen abzudecken. Grundsätzlich bevorzuge ich es aber eher weniger Themen, dafür aber etwas mehr ins Detail zu gehen um diese dann auch wirklich zu verstehen.

Security: Recording L6/7 schauen

- workflows und unitTests gestalten, wie aufbauen und von wo kriegt man die infos was da rein muss. Das WISO hast du klar rüber gebracht aber das umsetzen... PS: bei der frage 1 weiss ich nicht was das alles beinhaltet, sorry (da vertraue ich deinen erfahrungen)

Hallo Dominik, Ich denke es würde Sinn machen, wenn wir die eine oder andere Übung mit etwas mehr "Anleitung" anschauen könnten. Somit wird die Basis etwas stabiler. (hast Du oben in dieser Umfrage erwähnt)

Ich würde gerne Docker im Zusammenspiel mit Github Action nochmals durchnehmen.

⏰ bilal

- Technischemöglichkeiten aufzeigen und wie man dinge richtig löst... Datenbank mit docker und http - Du sagst immer mann soll nichts neu erfinden, es gibt schon alles besser als man es selber machen könnte, aber wo? Wie geht man dabei vor..? (Login, User-Rechte...)

Software Design Pattern. z.B. Strategy-, Observer-, Factorpattern usw.

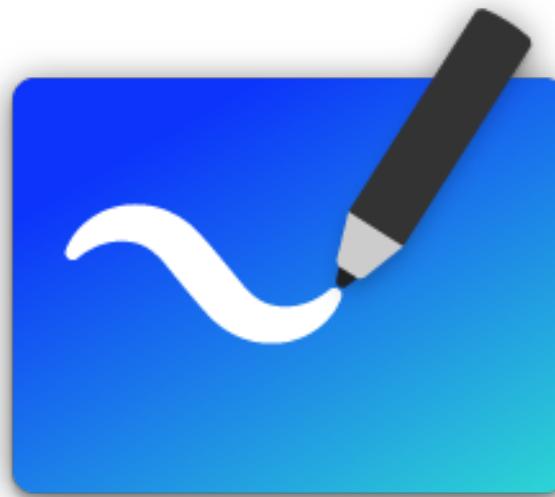
Transferaufgabe SWT

– Integration Testing

Transferaufgabe SWD

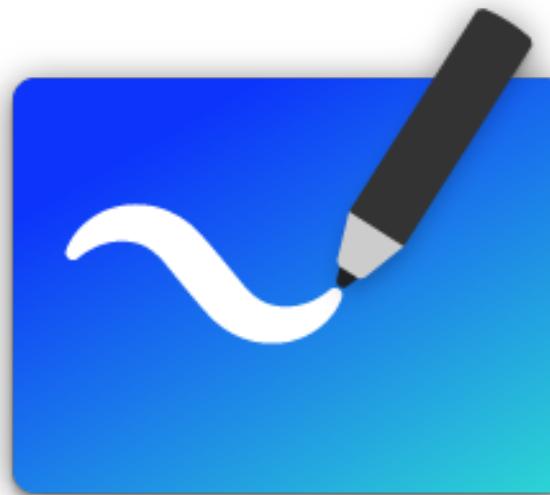
– Architektur

Zusatz: Wie lösen Workflows aus?



Controller

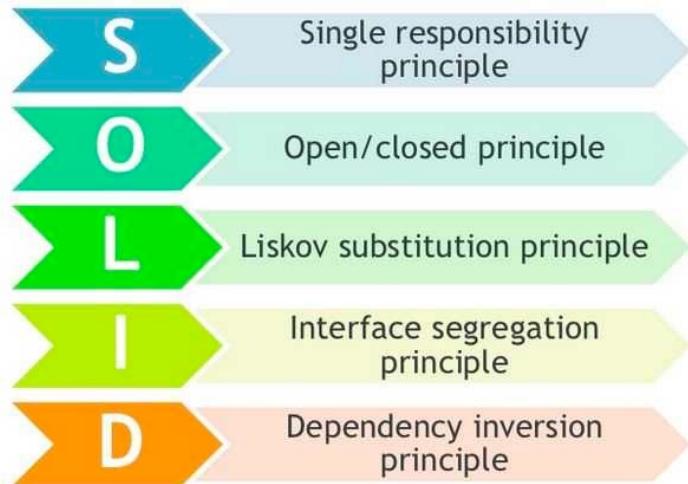
im Kontext Spring Web und als Teil der API



Design Considerations

- Compatibility - With others or an older self
 - Extensibility - Adding capabilities without major changes
 - Maintainability - Documentation, DevOps, Transparency, clean
 - Modularity - isolated components, exchangeable, division
 - Reliability - functions under load for a specific time
 - Reusability - Don't repeat yourself DRY, reuse in other designs
 - Robustness - Operate under stress / faulty data, fail graceful
 - Security - Withstand hostile acts and influence, keep data safe
 - Usability - Emotions, help, experience, journeys
-
- Design != Architektur, jedoch ist klar
 - Ungeeignete/schlechte Architektur kann kein gutes Design fördern
-
- Considerations sind in «Design Prinzipien» **SOLID** und **GRASP** verankert

kennen wir
von
Architektur



GRASP

GRASP Principles

- Expert
- Creator
- Low Coupling
- High Cohesion
- Controller
- Polymorphism
- Pure Fabrication
- Indirection
- Controlled Variation

Randnotiz (nicht Teil unseres Modules)

<https://12factor.net/>

Design “principles” for apps
not only for code

Twelve-Factor Applications		
1. CODEBASE One codebase tracked in SCM, many deploy	2. DEPENDENCIES Explicitly declare isolate dependencies	3. CONFIGURATION Store config in the environment
4. BACKING SERVICES Treat backing services as attached resources	5. BUILD, RELEASE, RUN Strictly separate build and run stages	6. PROCESSES Execute app as stateless processes
7. PORT BINDING Export services via port binding	8. CONCURRENCY Scale out via the process model	9. DISPOSABILITY Maximize robustness & graceful shutdown
10. DEV/ PROD PARITY Keep dev, staging, prod as similar as possible	11. LOGS Treat logs as event stream	12. ADMIN PROCESSES Run admin / mgmt tasks as one-off processes

Design patterns

“Design patterns are typical solutions to common problems in software design. Each pattern is like a blueprint that you can customize to solve a particular design problem in your code.”

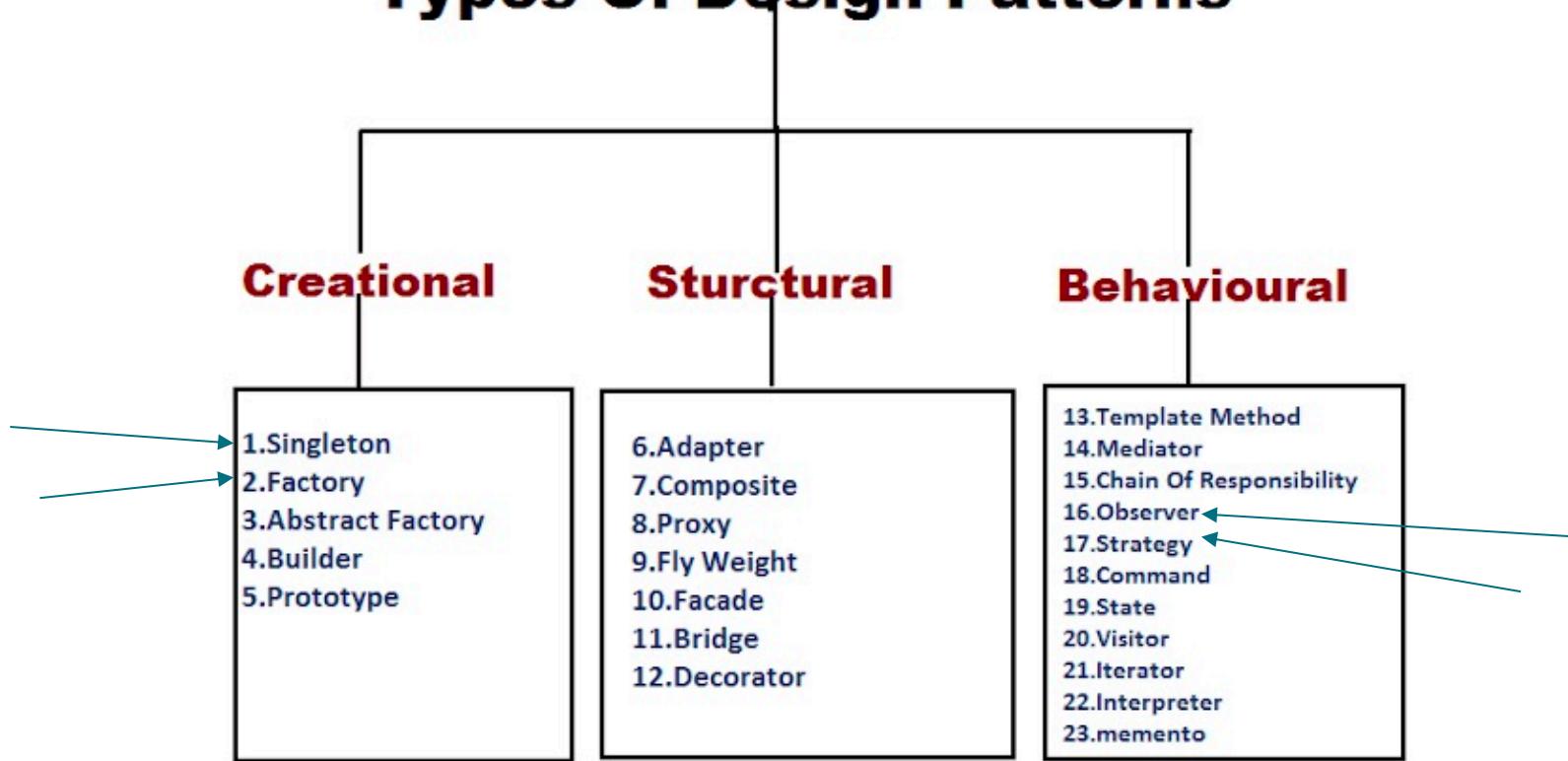
Design Guru

Design patterns - Anti patterns

An anti-pattern is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive.

[Wikipedia](#)

Types Of Design Patterns



Auswerten

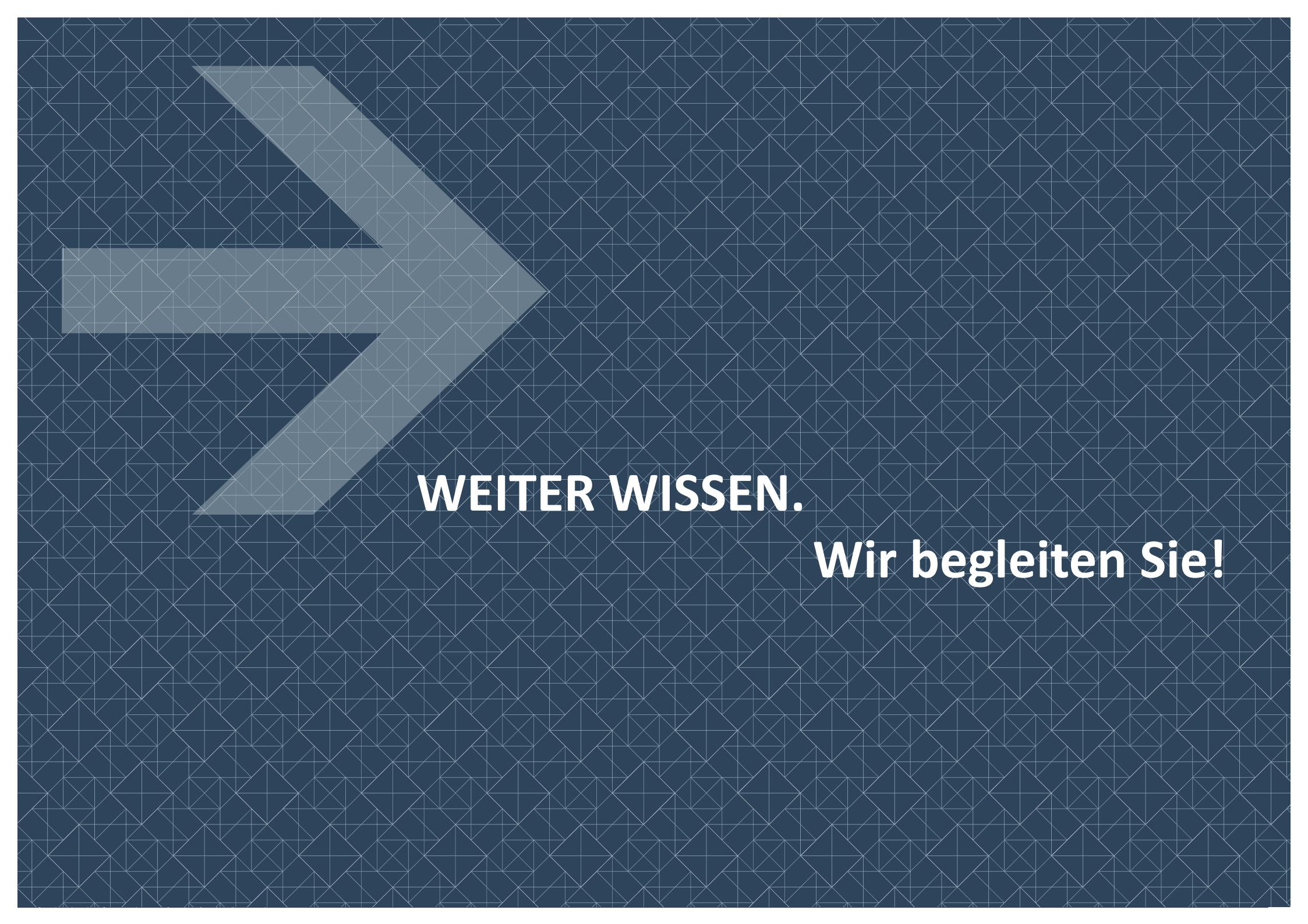
Zielkontrolle

AUFTAG
⌚ 5min

<https://forms.office.com/r/PN7a66WvmH>

🔒 ABB Login





WEITER WISSEN.

Wir begleiten Sie!